
Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática (DETI)

Ricardo José da Silva Correia Casaleiro

Sala de Espectáculos Virtual

Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática (DETI)

Ricardo José da Silva Correia Casaleiro

Sala de Espectáculos Virtual

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica dos Prof. Doutores Paulo Dias e Guilherme Campos, ambos Professores do Departamento de Electrónica, Telecomunicações e Informática (DETI).

o júri

presidente

Doutor Alexandre Manuel Moutela Nunes da Mota
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Doutor Diamantino Rui da Silva Freitas
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Doutor António Guilherme Rocha Campos
Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

palavras-chave Auralização, HRTF, Método das Imagens Virtuais, Realidade Virtual, Reverberação.

resumo

O trabalho apresentado nesta tese tem como objectivo a auralização em ambientes virtuais.

Para esse efeito foi desenvolvida a biblioteca *3DLIB* que permite gerar som 3D em tempo real num ambiente simples baseado em óculos de realidade virtual (*Head Mounted Display - HMD*), auscultadores e um sensor de detecção da orientação da cabeça do utilizador (*head tracking*). Ambas as informações – visual e auditiva – são simultaneamente actualizadas em tempo real.

O algoritmo de auralização recorre a HRTF (*Head-Related Transfer Functions*), escolhidas tendo em conta a orientação da cabeça, para processar o som directo e as primeiras reflexões. O ruído (*clicking*) que ocorre nas transições entre HRTF é removido através de interpolação.

As primeiras reflexões são determinadas pelo método das imagens virtuais: o contributo de cada reflexão é obtido por processamento do som directo tendo em conta os coeficientes de atenuação dos materiais das superfícies.

Os materiais são especificados num programa de correcção acústica de salas desenvolvido num projecto anterior. Este permite associar coeficientes de absorção a cada polígono de um modelo 3D. Esta informação é usada para calcular os tempos de reverberação por banda usados na simulação.

Foram estudados e implementados vários algoritmos de reverberação entre os quais se destacam a resposta impulsional sintética, a FDN (*feedback delay network*) e o algoritmo *multirate*. A FDN permite simular um tempo de reverberação médio enquanto os restantes permitem simular tempos de reverberação distintos por banda.

No sentido de validar os algoritmos implementados, foram desenvolvidas demonstrações e aplicações que permitiram a realização de testes perceptuais preliminares.

keywords Auralisation, HRTF, Image Source method, Virtual reality, artificial reverberation.

abstract

The main objective of the work presented in this thesis is auralisation in virtual environments.

For this purpose the *3DLIB* library was developed which allows 3D sound to be generated on a simple virtual environment based on a Head Mounted Display (HMD), headphones and a head tracking device. Both visual and aural information are updated simultaneously in real time.

The auralisation algorithm uses HRTF (*Head-Related Transfer Functions*), selected according to head orientation, to process the direct sound and early reflections. Clicking artifacts caused by HRTF transitions are removed trough interpolation.

Early reflections are computed using the image-source method: the contribution of each reflection is obtained by processing the direct sound taking into account the absorption coefficients of surface materials.

The materials are specified using an acoustic correction program developed in a previous project. With this software it is possible to assign absorption coefficients to each polygon of a 3D model. Octave-band reverberation times (RT) are then computed using this information.

Several reverberation algorithms were considered and implemented, such as the Synthetic Impulse Response, FDN (feedback delay network) and multirate algorithm. With the FDN method it is possible to simulate an average RT while the others allow octave-band calculations.

In order to validate the implemented algorithms, a number of demonstrations and applications were developed. These allowed preliminary perceptual tests.

Conteúdo

Ilustrações

1 Introdução

A *realidade virtual* (*RV*) é uma área de crescente importância, com uma gama de aplicações também ela crescente: desde os jogos de computador e instalações artísticas multimédia ao estudo e divulgação do património arquitectónico e arqueológico, passando, por exemplo, pelo desenvolvimento de projectos de urbanismo, arquitectura, decoração ou promoção turística.

Naturalmente, os esforços para criação de ambientes de *RV* têm-se centrado sobretudo na vertente visual. Ora, não obstante o predomínio indiscutível da visão, a nossa percepção da realidade passa por mais quatro sentidos. A criação de ambientes virtuais cada vez mais convincentes exige que também eles sejam considerados. O mais importante – pelo menos entre aqueles que não envolvem contacto físico directo – é, sem dúvida, a audição. Este trabalho coloca ênfase precisamente na imersão auditiva, visando recriar em simultâneo, de forma articulada e em tempo real, os ambientes visual e sonoro de uma sala [Begault, 1994]. Com isto pretende-se criar uma *sala de espectáculos virtual*, pois o *espectador* (pelo menos na acepção tradicional do termo), não envolve qualquer uso significativo dos sentidos do tacto, paladar ou olfacto.

Existem vários equipamentos direccionados para ambientes de *RV*. Esses equipamentos tentam estimular os sentidos do ser humano e a forma como ele interage com o mundo. Existe hardware específico para ambientes de *RV* que facilita essa simulação. Neste projecto foi utilizado um ambiente para aplicações de Realidade Virtual e Aumentada disponível no IEETA baseado numa câmara sem fios, óculos de realidade virtual (*Head-Mounted Display* – HMD – ilustrado na Figura 1) e um sensor de orientação InterTrax (Figura 2) com 3 graus de liberdade.

Figura 1 - Head Mounted display

Figura 2 - Sensor InterTrax

A descrição técnica dos *HMD* e do sensor InterTrax encontra-se no anexo 1.

2 Modelação acústica

Este capítulo apresenta alguns conceitos da Acústica e da Psicoacústica que estão na base do trabalho desenvolvido. Trata-se da forma como os sons se propagam num determinado ambiente físico e que aspectos (e como) influem na percepção desses sons.

2.1 Auralização

A recriação artificial de um ambiente sonoro é chamada *auralização*. Consiste em reproduzir (normalmente através de auscultadores) os estímulos sonoros que atingiriam os tímpanos do ouvinte se ele se encontrasse na sala que se pretende simular, para uma dada posição da(s) fonte(s) e do próprio ouvinte.

O comportamento acústico de uma sala é descrito pela sua *resposta impulsional* (*RIR - Room Impulse Response*) – dita *binaural* no caso de se considerar um par de sinais (um para cada ouvido). A auralização baseia-se na convolução da *RIR* binaural com gravações anecóicas do som emitido pela fonte.

Existem hoje processos experimentais rigorosos para medir *in situ* a *RIR* binaural de uma sala, o que torna trivial a auralização com *RIR* medida. Porém, calcular uma *RIR* com base num modelo em tempo real (e conseguir uma auralização convincente nestas condições) é um desafio muito complexo. Para isso é necessário compreender de que forma o som é percepcionado pelo ser humano, tendo em conta o ambiente que o rodeia.

2.2 Propagação do som

Quando uma fonte pontual emite um som, ele propaga-se em ondas esféricas. Assim, a energia da onda reparte-se por uma superfície $4\pi d^2$, sendo d a distância à fonte. Essa superfície aumenta à medida que a onda se propaga pois $d=ct$ onde c é a velocidade do som. Isto quer dizer que a intensidade do som que chega ao ouvinte vai diminuindo em função da distância a que este se encontra da fonte sonora segundo uma lei quadrática inversa *(inverse square law*). Por exemplo, ao aumentar a distância para o dobro, a energia é reduzida por um factor de 4 (conforme ilustra a Figura 3) [Howard el al, 1996].

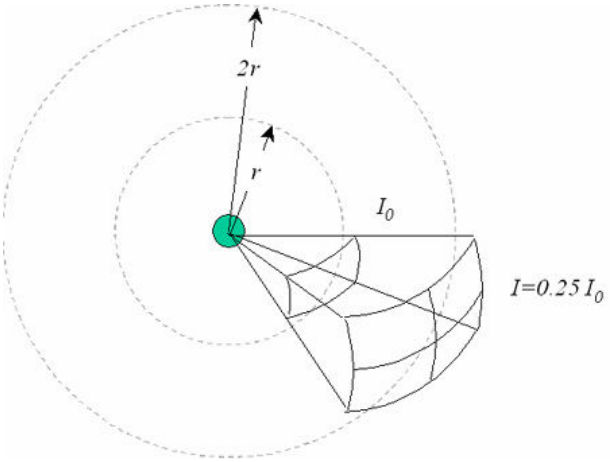


Figura 3 -Propagação da energia sonora

Seja E_0 a energia da onda sonora num ponto à distância de 1 m da fonte. Então a energia E a uma distância qualquer d é dada por:

$$E = \frac{E_0}{d^2} \qquad (2.1)$$

Como a energia da onda sonora é proporcional ao quadrado da sua amplitude [White, 1975], resulta:

$$A = \frac{1}{d} A_0 \qquad (2.2)$$

Em que d é a distancia entre o ponto e a fonte sonora e A_0 a amplitude inicial a um metro de distância. A amplitude da onda sonora é assim inversamente proporcional à distância percorrida. Note-se que todo este raciocínio parte da premissa que a propagação ocorre em espaço aberto (*free-field),* ou seja, não é perturbado por nenhum objecto próximo, como uma parede por exemplo. Na vida real, estamos quase sempre rodeados de objectos ou dentro de espaços fechados, com os quais as ondas sonoras interagem (fenómenos de reflexão, absorção, refracção difracção, interferência). Na Figura 4 é possível visualizar um pequeno exemplo de como as ondas sonoras interagem com o meio. No exemplo dado é possível ver uma reflexão de primeira

e outra de segunda ordem (a ordem consiste no número de reflexões que a onda sonora efectua até chegar ao ouvinte), juntamente com o som directo.

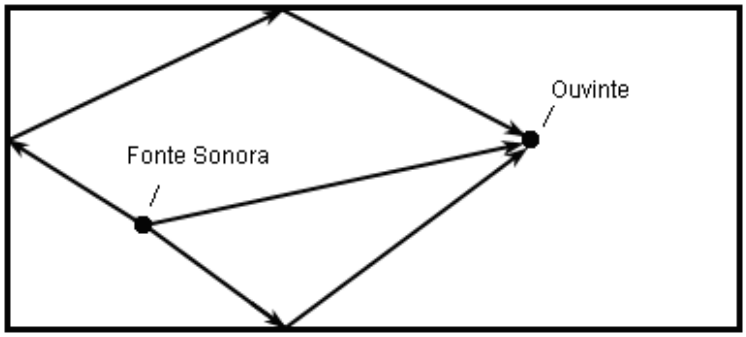


Figura 4 - Exemplo de som directo e de algumas reflexões

2.3 Percepção Sonora Espacial

Um aspecto particular da percepção humana do mundo consiste no facto de o ser humano ter noção da posição e da distância a que um som especifico se encontra no espaço pela forma como é percepcionado auditivamente. Isso é facilitado não só por, *a priori*, cada ouvinte ter conhecimento do ambiente que o rodeia, sons que nele possam existir e as suas possíveis localizações, mas também pela interacção com o mundo. Cada vez que o ouvinte se movimenta recolhe novas informações sobre o ambiente que o rodeia. Pode assim actualizar constantemente o modelo cognitivo do mundo, inserindo assim o som percepcionado num contexto espacial. Se por algum motivo deixa de poder interagir com o mundo que o rodeia, a sua capacidade para determinar a posição das fontes sonoras diminui drasticamente [Begault, 1994].

Com auscultadores as pistas de localização fornecidas pelo movimento do utilizador perdem-se [Kendall, 1995], a não ser que se utilize processamento capaz de corrigir o som em tempo real em função desse mesmo movimento. Isto implica sistemas sofisticados de seguimento (*tracking*) do ouvinte que serão discutidos adiante em mais detalhe.

As secções seguintes abordam importantes conceitos relacionados com a percepção auditiva e capacidade humana para localizar as fontes sonoras.

2.4 *Head Related Transfer Function* (HRTF)

A forma como as ondas sonoras atingem os tímpanos de uma pessoa é afectada pela interacção das ondas sonoras originais com o tronco, a própria cabeça, e, em particular, os canais auriculares. O resultado dessas interacções pode ser medido e capturado sob a forma de um sinal normalmente denominado na literatura como "função de transferência relacionada com a cabeça" (*Head Related Transfer Function – HRTF)*.

A complexidade da interacção das ondas sonoras com o corpo humano torna as HRTF de cada ouvido extremamente dependentes da direcção do som. É a diferença entre as HRTF de cada ouvido que confere ao ser humano a capacidade de localização espacial das fontes sonoras. As pistas principais encontram-se nas diferenças no instante de chegada (*interaural time diference* – ITD) e de intensidade (*interaural intensity diference* – IID) dos sons em cada ouvido.

A posição do som em relação ao centro da cabeça é dado por um vector geralmente expresso através dos ângulos 'azimute' e 'elevação' e um valor escalar correspondente à distância a que o som se encontra (Figura 5).

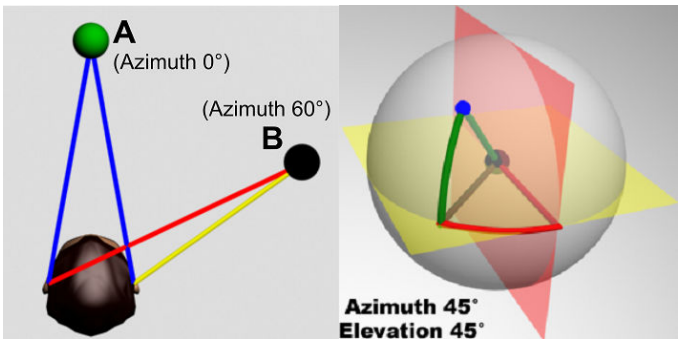


Figura 5 - Exemplos dos ângulos azimute e elevação

Cada ser humano tem características corporais únicas determinando assim a distinção entre as HRTF. Cada pessoa está naturalmente treinada para usar as suas próprias HRTF. Se 'usar' as de outra pessoa, a sua capacidade de localização é afectada, mas, após um período de adaptação pode até melhorar [Kendall, 1995].

2.5 Reverberação

A maior parte da vida do ser humano é vivida em ambientes reverberantes. Independentemente de o ouvinte estar a assistir a um concerto num auditório ou ao ar livre, o som percepcionado é sempre acompanhado por reflexões do som vindas de diversos pontos. Se o intervalo de tempo entre som directo e 1ª reflexão (entre reflexões) for superior a um certo valor (da ordem dos 20 ms) eles são percebidos como sons distintos (ecos). Se for inferior, isso não acontece. Em vez disso, os sons são percebidos como um só som de diferentes características (timbre, amplitude, espacialidade). Este efeito é chamado reverberação.

A reverberação é um efeito quase obrigatório na música causado naturalmente pela acústica natural das salas ou introduzido artificialmente em estúdio. Sem reverberação torna-se muito seca, perdendo grande parte da sua vivacidade. Por outro lado, muita reverberação ou o tipo errado de reverberação pode fazer com que uma boa 'performance' musical seja pouco inteligível, perdendo assim o seu encanto.

O processo da reverberação começa quando é iniciada a reprodução de um som numa sala. A onda sonora expande-se radialmente, distanciando-se da fonte, chegando até às paredes e outras superfícies da sala onde a energia da fonte é absorvida e reflectida. Tecnicamente, todo o som que é reflectido denomina-se reverberação [Everest, 2001].

Assumindo que existe um caminho directo entre a fonte e o ouvinte, este irá ouvir primeiro o ***som directo***, seguido das suas ***primeiras reflexões***. Após algumas dezenas de ms, a densidade de reflexões torna-se bastante elevada e cria o que é denominado por ***reverberação tardia***. Esta é independente da posição sonora e fornece informação espacial sobre o ambiente onde o ouvinte se encontra. Na Figura 6 é possível ter uma noção temporal mais clara sobre as distintas fases da percepção do som.

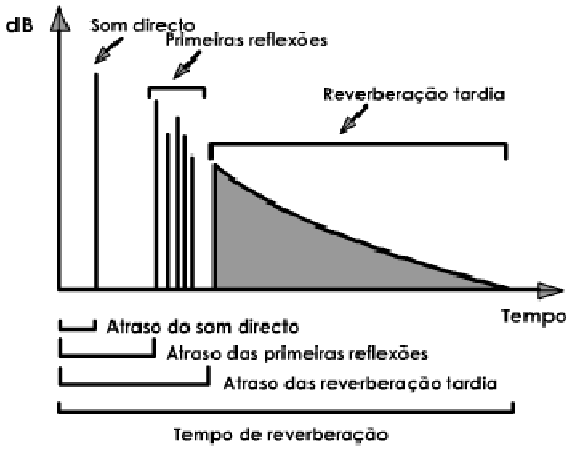


Figura 6 - Fases distintas da percepção do som

Num campo sonoro difuso, a energia perdida devido a absorções dos materiais é proporcional à densidade de energia do campo sonoro. Por isso, a reverberação difusa tem uma atenuação exponencial em relação ao tempo. O tempo necessário para que o som seja atenuado em 60 *dB* em relação ao nível inicial é definido como tempo de reverberação (RT_{60}) [Everest, 2001].

2.5.1 Aspectos perceptuais das Primeiras Reflexões

Consideremos um ambiente sonoro constituído pelo som directo e por uma reflexão gerada por esse som. Quando o som e a reflexão se encontram à frente ou atrás do utilizador e o atraso da reflexão é maior que 80 milissegundos, essa reflexão é percebida como um eco distinto do som directo, caso contenha energia suficiente para ser percepcionada. À medida que o atraso se torna menor, o som e a reflexão fundem-se num só som mas com uma coloração tonal, atribuída ao cancelamento ou reforço entre os dois sinais a determinadas frequências. O limiar de atraso e ganho correspondente a diferentes percepções depende do som emitido pela fonte sonora.

A informação espacial do som pode ser alterada por uma reflexão se a mesma vier de uma direcção lateral ao som directo. Para atrasos bastaste pequenos (menores que 5 milissegundos), o som proveniente da reflexão pode fazer com que a posição aparente do som se altere. Para atrasos maiores pode aumentar o tamanho aparente da fonte sonora, dependendo das frequências que estão a ser emitidas, ou pode dar a sensação que o utilizador está a ser rodeado pelo som.

Existem vários termos para denominar as sensações espaciais atribuídas às reflexões laterais tais como *'spaciousness', 'spatial impression', 'envelopment','apparent source width'*.

Segundo *Beranek* [Beranek, 1954], a impressão espacial ('*spatial impression*') é um atributo desejável na reverberação. Admite-se que as reflexões laterais alteram o carácter espacial do som devido a influenciarem directamente os mecanismos de localização do sistema auditivo. A presença de energia lateral contribui para diferenças interaurais (diferença entre o som de chegada a cada ouvido) que não ocorreriam apenas com o som frontal [Gardner, 1998].

2.5.2 Aspectos perceptuais da reverberação tardia

A reverberação tardia é um factor de relativa importância, na área musical. É através do seu uso que se obtém vivacidade na música, caso contrário a música torna-se 'seca'. O efeito da reverberação na música ainda não é totalmente compreendido, sendo uma área ainda em estudo [Everest, 2001].

A reverberação é também extremamente importante em salas de conferência, podendo o discurso ser inteligível em salas com um tempo de reverberação elevado. Em [Everest, 2001] existe um exemplo que ilustra este problema através da propagação de energia da palavra inglesa *"back",* em que a primeira parte '*ba*' é bastante impulsiva, enquanto na segunda '*ck*' existe uma quebra da energia, quando comparada com a primeira. Nestes casos é importante saber precisamente o tempo de reverberação da sala em questão porque se o tempo de reverberação for bastante elevado pode prejudicar a percepção do discurso, mascarando algumas sílabas e tornando-o ininteligível (problema ilustrado na figura 7).

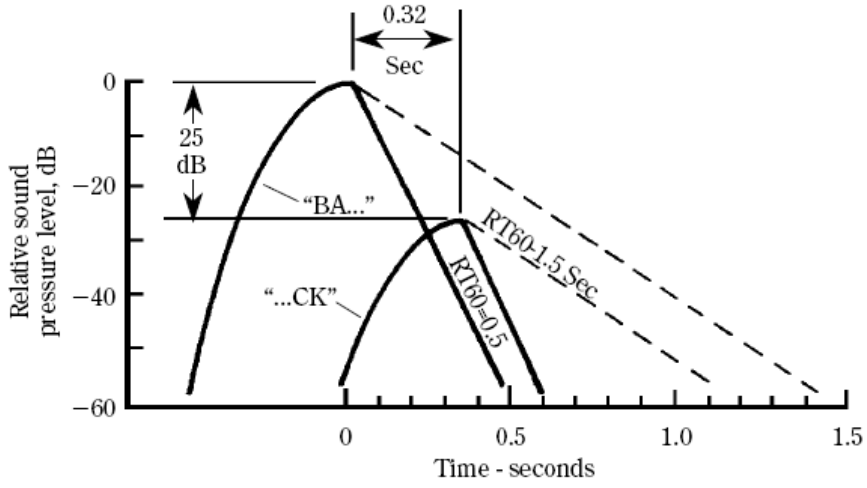


Figura 7 - Exemplo do efeito da reverberação num discurso (Fonte: [Everest, 2001])

3 Auralização

Nesta secção está descrito o algoritmo de auralização implementado, baseado em HRTF. Nesta fase assume-se que o som é emitido em espaço aberto, ou seja, é apenas considerado o som directo. As primeiras reflexões e a reverberação tardia são analisadas no capítulo 4. Encontram-se também descritos os passos para implementar o algoritmo em tempo real.

3.1 Algoritmo de auralização

As HRTF são usadas para simular a posição específica de uma fonte no espaço. Isso é conseguido através da convolução do som de saída escolhido para a fonte sonora e as HRTF da posição específica a simular (conforme ilustra a Figura 8). O som de entrada (x(t)) é duplicado para assim ser aplicado ao par de HRTF pretendido. Por fim, o resultado é apresentado ao utilizador através de auscultadores. Foram efectuadas várias experiências em MATLAB para se obter um maior conhecimento sobre a implementação e problemas do algoritmo de auralização, Essas experiências encontram-se descritas no anexo 2.

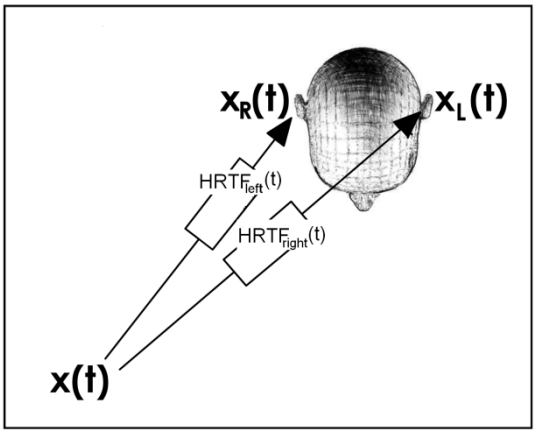


Figura 8 - Algoritmo base de auralização

3.2 HRTF utilizadas

Existe um conjunto de pares de HRTF razoavelmente completas e usado em experiências [Gardner, 1999] disponível para uso gratuito. Essas HRTF foram recolhidas através de um '*KEMAR dummy head microphone*' e encontram-se disponíveis na página do *Massachusetts Institute of Technology*(MIT) [HRTF, 2008]. Foram recolhidas várias amostras para elevação entre -40 e 90 graus e azimute entre 0 e 180 graus. As amostras foram apenas recolhidas com o ouvido esquerdo do boneco *KEMAR* (Figura 9). As HRTF para ângulos simétricos em relação ao plano mediano (que divide o hemisfério esquerdo e direito) são iguais, com a diferença que os canais são invertidos (a HRTF do ouvido esquerdo passa a ser a do direito e vice-versa). Para se obter as HRTF com valores de azimute no intervalo [180, 360] é necessário usar a HRTF simétrica do hemisfério direito e trocar o canal direito pelo esquerdo e vice-versa. Embora existam bastantes medições nas HRTF recolhidas, não existem medições para todos os ângulos possíveis. É possível obter uma aproximação de uma HRTF que não foi medida através da interpolação das HRTF mais próximas à pretendida. A Figura 10 exemplifica esse cálculo através do uso da interpolação bilinear. Em a) é possível ver a HRTF pretendida (HRTF de elevação 4 e azimute 3) e as HRTF mais próximas. O primeiro passo para a determinar consiste em determinar HRTF auxiliares para o ângulo azimute pretendido, interpolando as amostras dos azimutes mais próximos, para cada elevação (HI1 e HI2). O segundo passo consiste em interpolar as HRTF obtidas em b). O resultado é exemplificado em c).

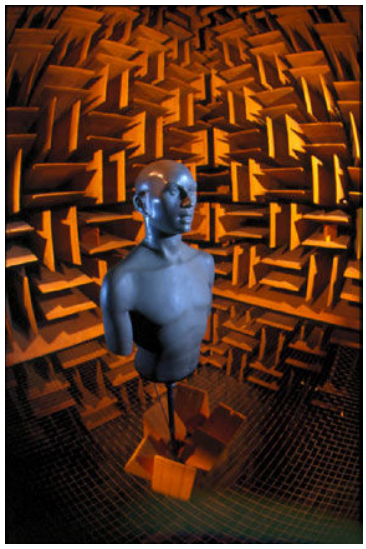


Figura 9 - Boneco KEMAR (fonte: [HRTF, 2008])

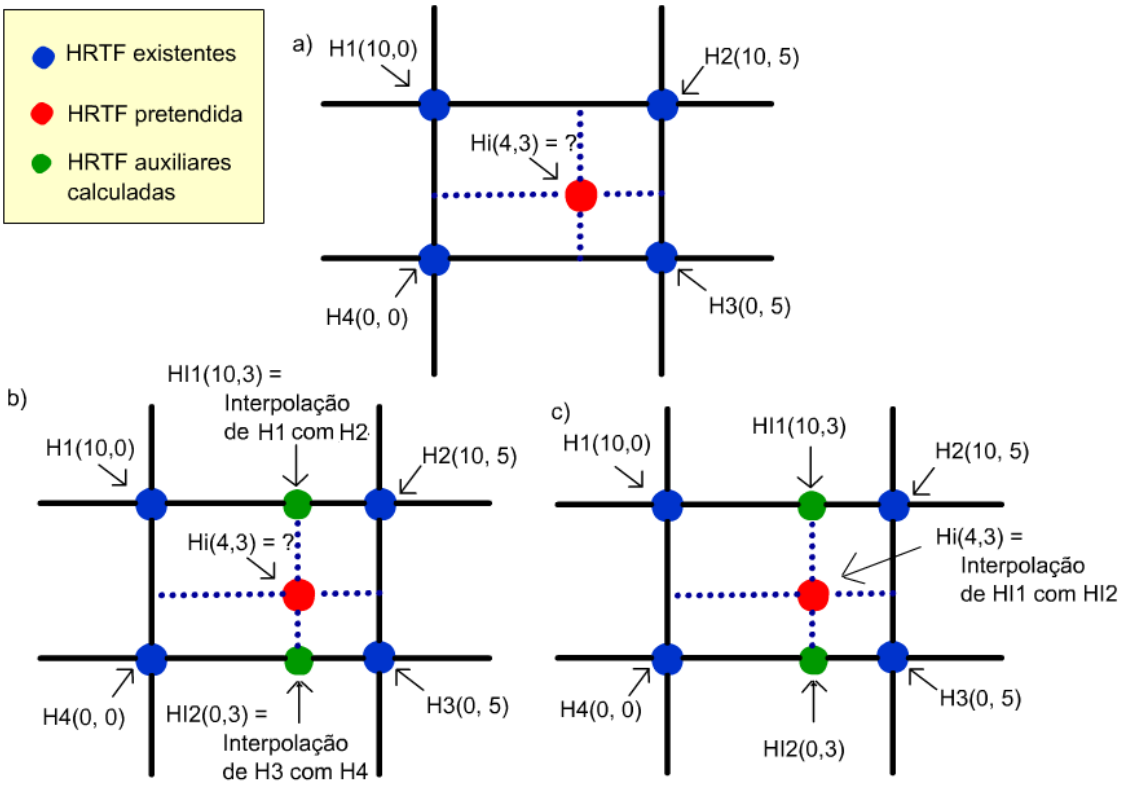


Figura 10 - Interpolações necessárias para obter a HRTF pretendida

3.3 Determinação dos ângulos azimute e elevação

Para se conseguir simular a posição de uma fonte sonora em relação a um ouvinte é necessário determinar o ângulo azimute e elevação tendo em conta a posição da cabeça do utilizador relativamente à fonte.

A orientação das câmaras de filmar, é normalmente definida por dois vectores: 'forward' e 'up'. O vector 'forward' é definido pela posição da câmara e seu ponto de foco. O vector 'up' é perpendicular ao vector 'forward' e aponta para o topo da câmara

A orientação da cabeça de um utilizador pode ser definida de forma idêntica (figura 11).

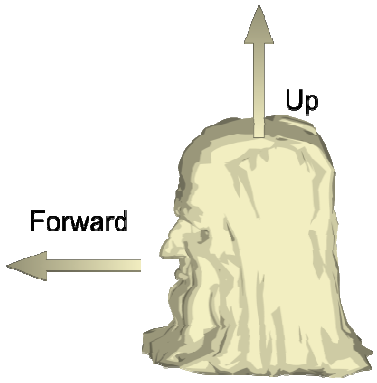


Figura 11 - Vectores *forward* e *up*

O utilizador pode girar a cabeça em três direcções distintas. Esses ângulos de rotação são normalmente denominados por pitch, yaw e roll (exemplificados na Figura 12). Quando alteramos um dos ângulos de rotação é necessário aplicar a rotação correcta aos vectores 'forward' e 'up'. Para isso é necessário calcular a matriz de rotação a aplicar a cada um. A matriz é calculada tendo em conta que as seguintes rotações são equivalentes:

- Yaw - Rotação no eixo YY;
- Pitch - Rotação no eixo XX;
- Roll - Rotação no eixo ZZ.

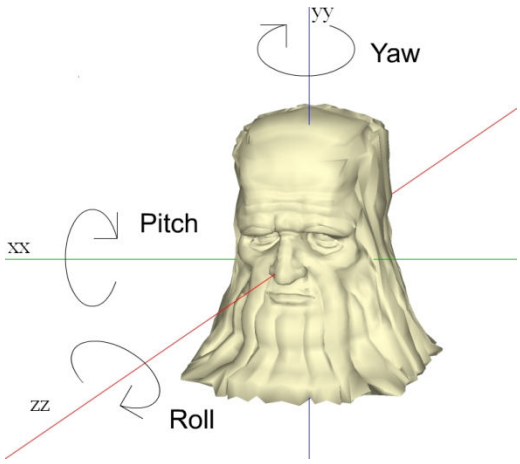


Figura 12 - ângulos de rotação

3.4 Auralização em tempo real

Como foi referido anteriormente, a percepção espacial sonora diminui drasticamente se o utilizador não puder interagir com o mundo que o rodeia. Assim, um factor importante para a criação de um ambiente acústico virtual é a capacidade de interacção que o utilizador tem perante o mundo. A resposta a essa interacção não pode ocorrer num tempo superior a 150 ms, caso contrário a capacidade de localização do utilizador decresce drasticamente [Wu e Mu, 1997]. Por consequência, todos os cálculos necessários para a implementação da auralização têm que ser efectuados num tempo inferior a 150 ms.

3.4.1 Implementação da auralização

Foram avaliadas várias ferramentas na criação do ambiente virtual. O ambiente de desenvolvimento escolhido foi o Microsoft Visual C++. Não existe nenhuma biblioteca predefinida em C++ que permita reproduzir sons. Para isso é necessário recorrer a bibliotecas áudio externas. Existem várias bibliotecas áudio disponíveis tais como o *FMOD Ex* [FMOD, 2008], *DirectSound* (incluído no *DirectX*) [DirectSound, 2008] e *PortAudio* [PortAudio, 2008], entre outras. Um requisito essencial para a escolha da era a possibilidade de se ter acesso às amostras de saída para as poder editar antes da sua reprodução. Após algumas experiências com algumas das bibliotecas acima mencionadas (*FMOD Ex e PortAudio*), até encontrar a mais adequada. Após a implementação dessas experiências optou-se pela utilização da biblioteca *PortAudio* por permitir uma maior liberdade na edição das amostras a ser reproduzidas. Para mais informação sobre a biblioteca *FMOD* consultar o anexo 1.

A *PortAudio* (*PA*) não tem a capacidade de gerar ambientes sonoros 3D automaticamente. Para conseguir gerar um ambiente virtual é então necessário recorrer a HRTF. Foi criado um

pequeno programa para testar as funcionalidades do PA e verificar como possibilita implementar som 3D. Segue-se um resumo do trabalho de familiarização com o PA e os seus aspectos mais salientes.

a) Familiarização com o *PortAudio*

Os passos necessários para criar uma aplicação em PA são os seguintes:

- Escrever uma função de c*allback*.
- Inicializar a biblioteca PA e abrir um *stream* para a entrada/saída de áudio
- Iniciar a reprodução. A função de c*allback* criada irá ser chamada repetidamente pelo PA em *Background*
- Parar a reprodução do *Streaming*
- Fechar o *stream* e terminar o PA

A função de *callback* é uma função invocada pelo motor do PA sempre que são necessárias amostras áudio para o *buffer* de saída. Em cada execução é lido um número predefinido de amostras. Assim, caso o objectivo seja reproduzir um ficheiro de som, é necessário, dentro da *callback*, preencher o *buffer* de saída com essas amostras.

b) Criação de som 3D com o *PortAudio*

Como o PA não tem a capacidade de reproduzir sons 3D é necessário recorrer às HRTF. Como em C não existe nenhuma função predefinida que calcule a convolução de dois sinais, foi necessário implementá-la. O algoritmo usado na sua implementação foi baseado no algoritmo fornecido em [Orfanidis, 1996]. O algoritmo apenas converte a fórmula da convolução para uma função em C. Como o PA não tem suporte para a leitura de ficheiros de som é necessário usar outra biblioteca para esse efeito. A biblioteca escolhida foi a *libsndfile*, recomendada pela equipa do PA [Libsndfile, 2008].

A reprodução de sons no PA é efectuada através do preenchimento do *buffer* de saída. Esse buffer é um vector de tamanho predefinido que vai sendo preenchido em tempo real com cada segmento de som a reproduzir. O número de amostras em cada segmento é igual ao número de amostras que a função de *callback* processa cada vez que é executada.

A convolução entre cada segmento e as HRTF tem que ser efectuado dentro da *callback*. O método para o preenchimento do buffer encontra-se explicado em maior detalhe no ponto 3.5.

É necessário realçar que as HRTF são aplicadas ao som de entrada. Isso é conseguido por convolução de cada segmento do som de entrada com a HRTF específica. Tendo um segmento de som e um filtro com N e M amostras, respectivamente, o resultado da convolução entre

ambos conterá N+M-1 amostras. Se o nosso buffer de saída for igual ao número de amostras de cada segmento, ao preenchê-lo com o resultado da convolução vão sobrar M amostras. Essas amostras têm que ser adicionadas no início do buffer de saída seguinte.

c) Interacção com o utilizador

Após termos a reprodução do som 3D a funcionar é preciso fornecer meios para o utilizador interagir com o sistema. Inicialmente a interacção foi efectuada unicamente com o auxílio do teclado. O utilizador pode alterar os ângulos Azimute e Elevação. Mais tarde foi dada a possibilidade de o utilizador interagir com o ambiente virtual recorrendo ao sensor *InterTrax* (consultar anexo 1). Como o sensor *InterTrax* recebe os ângulos de orientação *Pitch*, *Yaw* e *Roll* é necessário convertê-los para elevação e azimute.

d) Criação de informação visual

Um dos problemas da experiência passava pela falta de informação visual. Decidiu-se por isso adicionar, com o auxílio do *VTK* [Ávila, 2004] (Ver anexo 1), uma esfera na posição da fonte sonora. A posição da esfera é alterada tendo em conta os valores lidos pelo sensor *InterTrax*.

e) Definição da latência de saída

Um ponto extremamente importante a ter em conta para produzir aplicações em tempo real é a latência (tempo entre o preenchimento do buffer de saída e a sua reprodução). Quando o utilizador interage com ambiente virtual, a posição das fontes sonoras altera-se. Isto tem que ocorrer com o mínimo de atraso possível. O valor máximo permitido para o atraso do som em relação à interacção do utilizador é de 150 ms para assim ser possível fornecer a resposta de uma interacção do utilizador com o mundo em tempo útil. No anexo 1, ponto 2, encontra-se descrito o método para alterar a latência de saída no PA.

3.4.2 Movimentação do utilizador

Quando o utilizador se movimenta, alterando assim um dos ângulos de rotação, é necessário actualizar a informação que ele tem sobre as fontes sonoras, ou seja, cada fonte virtual vai alterar a sua posição em relação ao utilizador. Para alterar a posição de uma fonte é necessário calcular o ângulo azimute e elevação que a fonte forma com a nova posição do utilizador e usar a HRTF correspondente no algoritmo de auralização. No entanto, se este processo for feito em tempo real vão ser gerados artefactos (*clicks)* no som em cada transição de filtros HRTF. Este fenómeno acontece devido ao filtro usado no algoritmo ser diferente entre duas iterações da função de *callback*. Na Figura 13 encontra-se exemplificado o método inicial de preenchimento do buffer de saída. Esse método apenas está correcto se a HRTF em cada iteração for igual; caso contrário são gerados os artefactos referidos anteriormente.

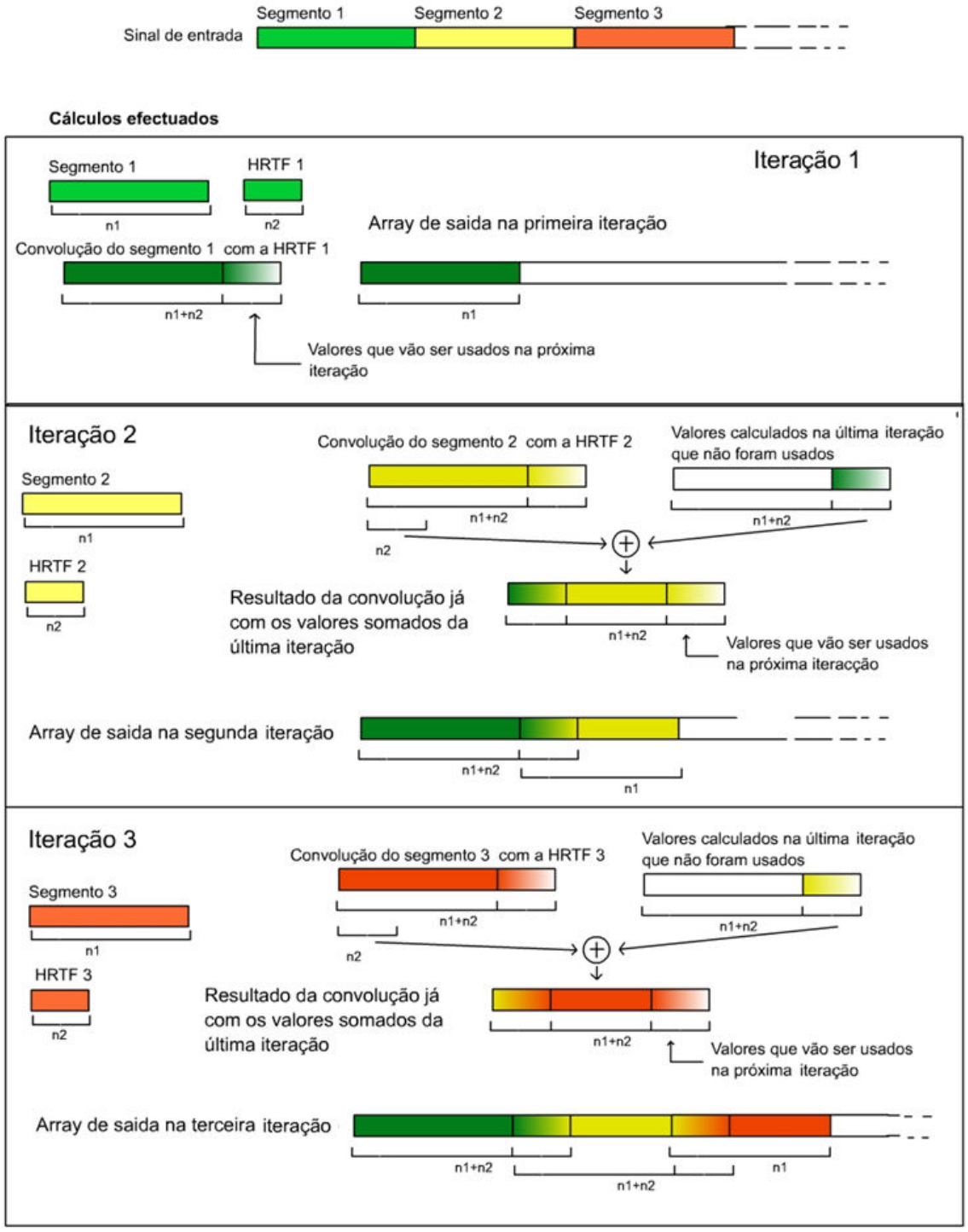


Figura 13 - Exemplo da resposta do algoritmo ao subdividir o sinal de entrada

Na Figura 14 é possível ver o ruído gerado em ambos os canais (esquerdo e direito) quando o ângulo azimute da fonte sonora em relação ao utilizador é alterado de 90º para 270º. Foi escolhido este par de ângulos por corresponder à situação mais desfavorável em termos de variação de ITD e IID, sendo o par que irá originar mais ruído quando ocorrer uma transição entre eles. À partida, essa transição nunca irá ocorrer num ambiente virtual, por existirem

sempre posições intermédias mas, ao eliminar o ruído na transição destas HRTF garante-se que não ocorrem clicks durante as outras transições cuja diferença angular será menor. Neste caso a transição ocorre a partir da amostra 270. A interferência gerada no som é visível entre as amostras 270 e 300 aproximadamente.

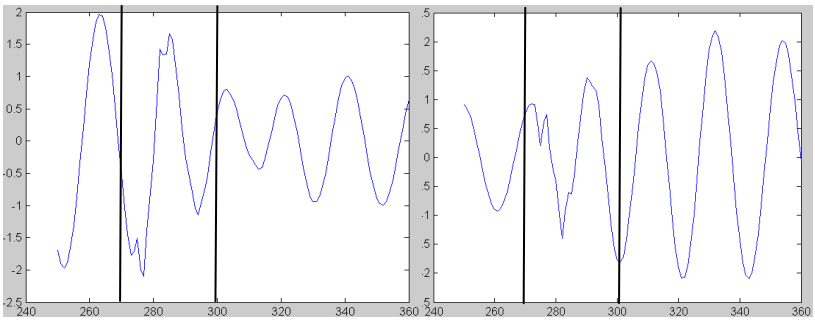


Figura 14 - Artefacto gerado no canal direito e esquerdo quando ocorre uma transição entre HRTF

A solução para eliminar os artefactos passou por duas fases. A primeira tentativa consistiu no seguinte: sempre que existir uma transição de HRTF calcular a convolução da nova HRTF com o último segmento de som usado com a HRTF antiga. De seguida, substituir as amostras que sobraram da última convolução calculada com a HRTF antiga pelas amostras calculadas com a nova HRTF (exemplificado na Figura 16). Verifica-se que o ruído deixa de existir no início de cada transição. No entanto, agora é criada uma quebra de amplitude no resultado de saída quando existe a transição entre as HRTF. Essa quebra pode ser visualizada na Figura 15.

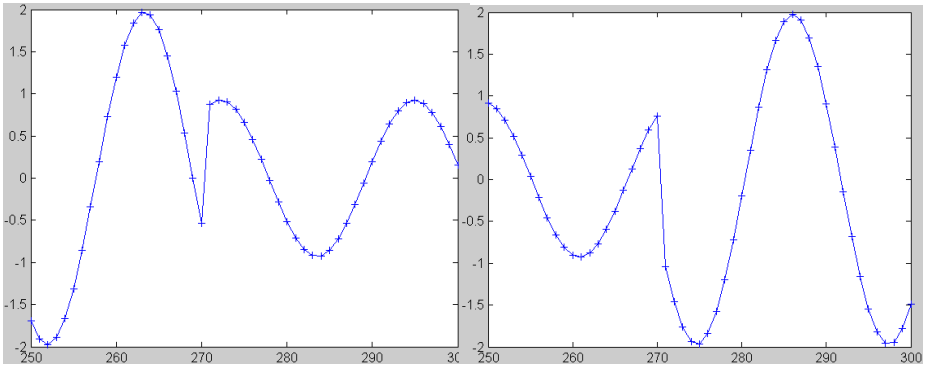


Figura 15 - Quebra na amplitude do som no canal direito e esquerdo quando ocorre uma transição entre HRTF

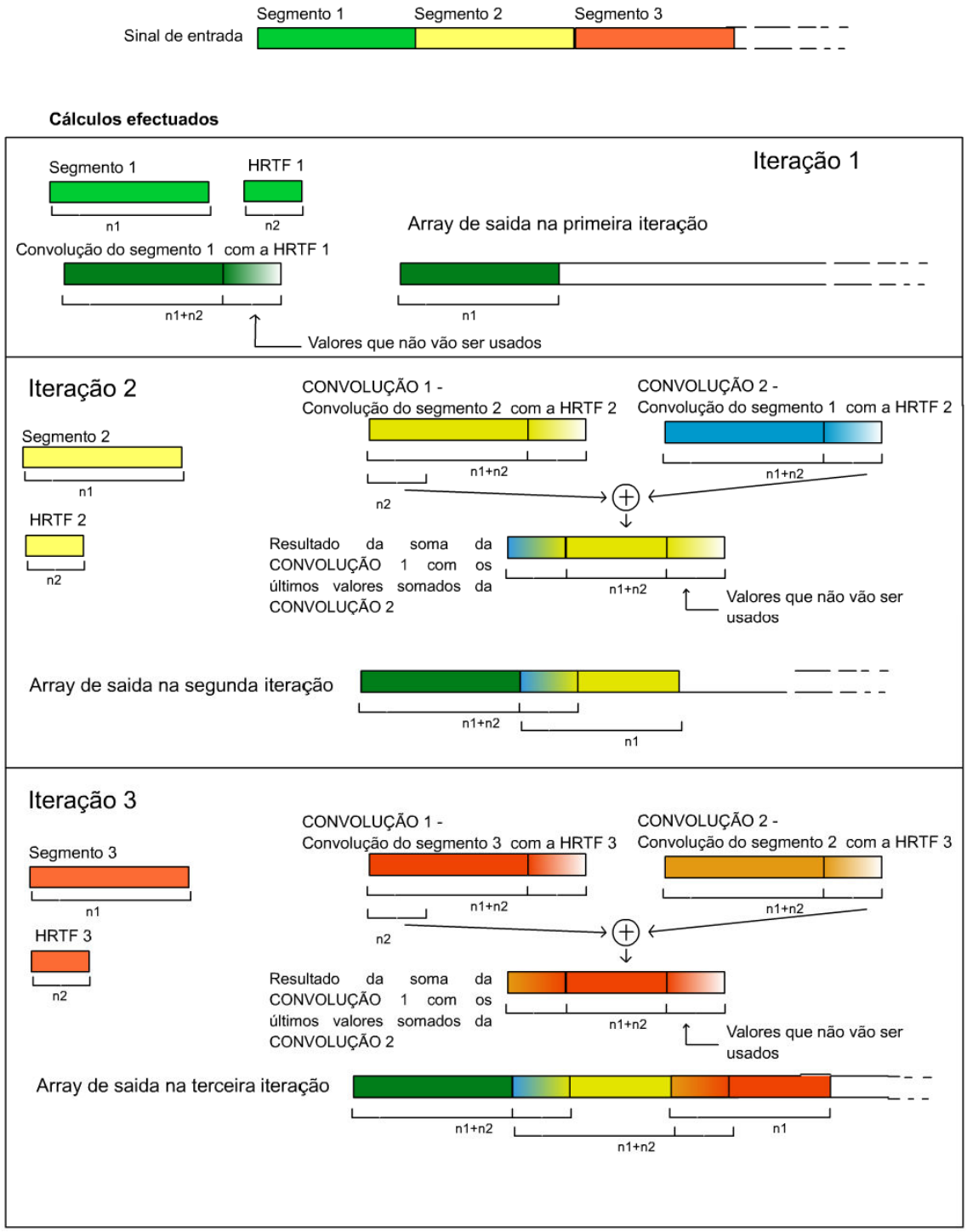


Figura 16 - Buffer de saída com a primeira tentativa de remoção do *clicking*

A segunda fase da solução consistiu na implementação de uma interpolação na transição das HRTF. No passo anterior substituíram-se as amostras que sobraram da última convolução calculada com a HRTF antiga pelas amostras calculadas com a nova HRTF. Agora usam-se

ambas as amostras e efectua-se uma interpolação linear de forma a obter uma transição suave entre os buffers com HRTF diferentes efectuando assim um *fade in* e um *fade out* entre os diferentes buffers em N amostras.

Na Figura 17 encontra-se o exemplo da forma como o sinal de saída é gerado para os dois primeiros segmentos.

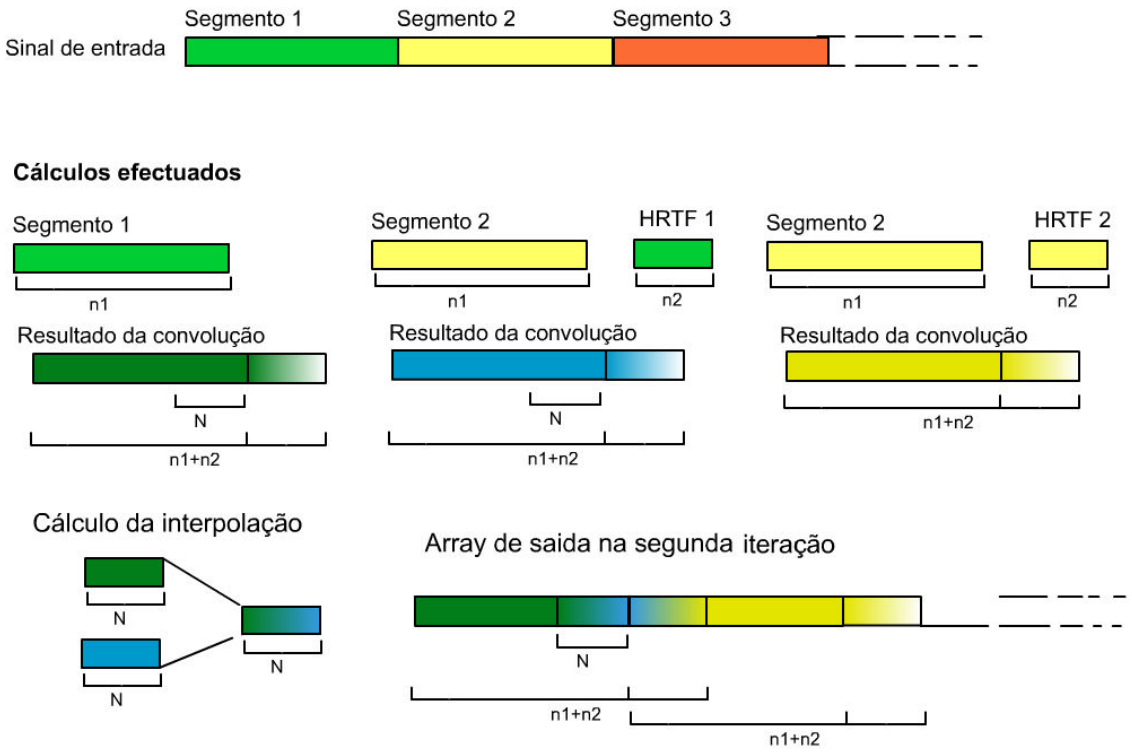


Figura 17 - Buffer de saída com a interpolação para remover o *clicking*

Ao aumentar o número de amostras usadas na interpolação, a intensidade de ruído diminui, sendo o mesmo quase imperceptível a partir de N = 70 amostras (ver Figura 18). Para efectuar este teste foi usado um buffer com 256 amostras, sendo o resultado válido para buffers com um maior número de amostras. No anexo 3 encontram-se imagens dos sinais obtidos para vários N (número de amostras usados na interpolação) distintos.

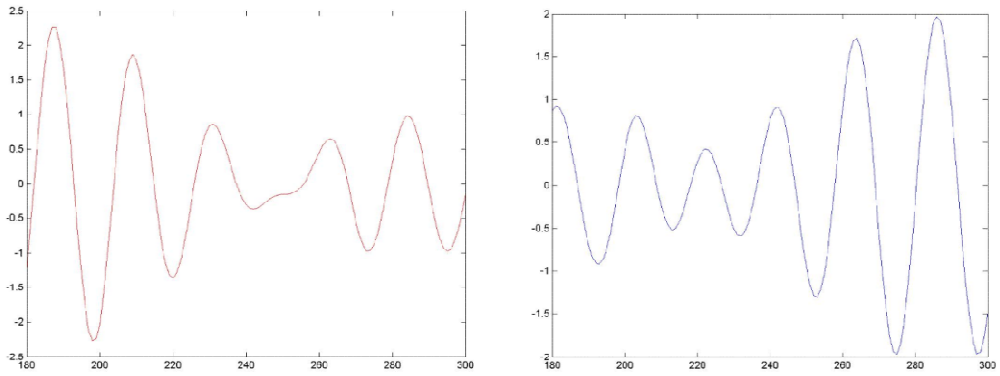


Figura 18 - Resultado final depois da interpolação

3.5 Implementação da auralização

O algoritmo para implementar auralização em tempo real está descrito no fluxograma seguinte

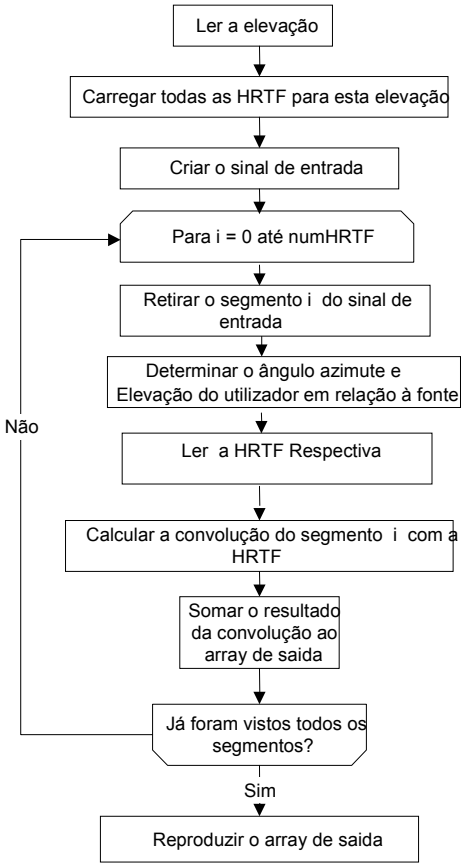


Figura 19 - Algoritmos de auralização em tempo real

Para efeitos de optimização é recomendável calcular a convolução de cada segmento com a HRTF no domínio da frequência. O tempo máximo (valores teóricos) que pode ser consumido pelo processador para reproduzir um som sem falhas encontra-se descrito no quadro seguinte.

Tamanho do buffer	128	256	512	1024
11025 Hz	11.61 ms	23,22 ms	46.44 ms	92.88 ms
22050 Hz	5.81 ms	11.61 ms	23.22 ms	46.44 ms
44100 Hz	2.9 ms	5.81 ms	11.61 ms	23.22 ms

Tabela 1 – Tempo de processamento máximo permitido para reproduzir um som

Os valores do quadro foram calculados da seguinte forma:

$$t_{Max} = \frac{t_{buf}}{freq_{Am}} \qquad (3.1)$$

onde t_{buf} é o número de amostras em cada buffer e $freq_{Am}$ a frequência de amostragem.

Calcular a convolução no domínio do tempo implica um peso computacional proporcional a N^2. Para o reduzir é habitual efectuar o cálculo calcular a convolução no domínio da frequência, que implica um peso computacional proporcional a NlogN.

Assim, calcular a convolução no domínio da frequência consiste em:

- Calcular a FFT (transformada de Fourier) do sinal de entrada
- Calcular a FFT do filtro (no nosso caso, uma HRTF)
- **Multiplicar os sinais** (pois a convolução no domínio do tempo corresponde à multiplicação no domínio da frequência).
- Calcular a FFT inversa do resultado da multiplicação

Como já foi dito anteriormente, quando é calculada a convolução de um sinal de N amostras com um filtro de M amostras, o sinal de saída vai ter N+M-1 amostras. Para se usar a convolução em frequência é necessário alterar o sinal de entrada e o filtro para que ambos tenham N+M-1 amostras. Na implementação criada isso é efectuado através da inserção de zeros em cada sinal de forma a se obter o número de amostras pretendidas. Esse método denomina-se por ***overlap-add*** e encontra-se ilustrado na Figura 20. Como se pode observar, o filtro de entrada (a) e o segmento de entrada (d) têm apenas 128 amostras. No entanto, como o sinal de saída da convolução de (a) e (d) vai conter 255 amostras, é necessário introduzir 127 zeros no final de ambos [Schafer et al, 1998].

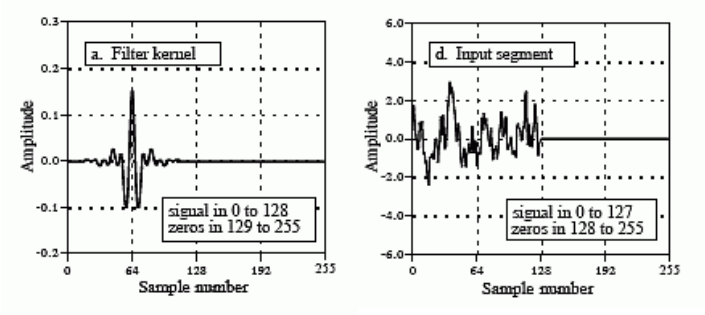


Figura 20 - Exemplo do método *overlap add* (Fonte: [Schafer et al, 1998])

A convolução nesta aplicação foi efectuada em duas fases. A primeira no domínio do tempo e a segunda no domínio da frequência.

Em C++, para se converter um ficheiro de som para o domínio da frequência pode-se recorrer á biblioteca FFTW ('*Fastest Fourier transform of The West*') [FFTW, 2008]. Essa biblioteca é

conhecida por ser a biblioteca *open-source* para cálculo de FFT mais rápida existente, competindo com as bibliotecas profissionais existentes no mercado. Na sua documentação existe um pequeno tutorial que explica o seu funcionamento (Consultar anexo 1 para mais detalhes).

A tabela 2 compara os tempos necessários para calcular a convolução no domínio do tempo e no domínio da frequência para diferentes valores de amostras por buffer (valores obtidos experimentalmente). O valor no domínio da frequência já considera o tempo gasto pela FFTW ao calcular a FFT e a IFFT.

	N = 256	N =512	N = 1024
Dom. Tempo	1,29 ms	2,56 ms	5,13 ms
Dom. Frequência	0,27 ms	0,41 ms	0,70 ms

Tabela 2

Analisando os valores do quadro anterior conclui-se que através da convolução no domínio da frequência é possível obter uma extraordinária melhoria de desempenho, traduzida na redução dos tempos de cálculo em cerca de 78,6 %, 83 % e 86.3 % para N = 256, 512 e 1024, respectivamente (valores obtidos comparando os valores no domínio do tempo na frequência).

4 Reverberação artificial

Este capítulo descreve o estudo que foi efectuado relativamente à reverberação do som. O objectivo consistia em analisar os algoritmos existentes e verificar quais os mais apropriados para gerar reverberação artificial cujo resultado fosse o mais natural possível. Como foi dito anteriormente, a reverberação pode dividir-se em duas fases: primeiras reflexões e reverberação tardia ou difusa. Cada fase tem métodos de simulação diferentes que irão ser especificados em maior detalhe nos pontos seguintes.

4.1 Programa de correcção acústica de salas

A reverberação de uma sala é determinada basicamente pela sua forma e pelos materiais que foram usados na sua construção. Através dessa informação é possível determinar analiticamente o tempo de reverberação de uma sala específica sendo possível usar esse valor para simular a reverberação de uma sala.

Num dos projectos do 5º ano do DETI no ano lectivo 2006-2007 foi desenvolvido um programa de correcção acústica de salas [Casaleiro et Al, 2007]. Este programa lê e manipula um modelo 3D de uma sala e possibilita a sua correcção acústica, através da definição dos materiais de cada superfície da sala. Após a definição dos materiais é possível determinar o tempo de reverberação em seis bandas de frequência, através da fórmula de *Sabine*. A Figura 21 ilustra a interface do programa.

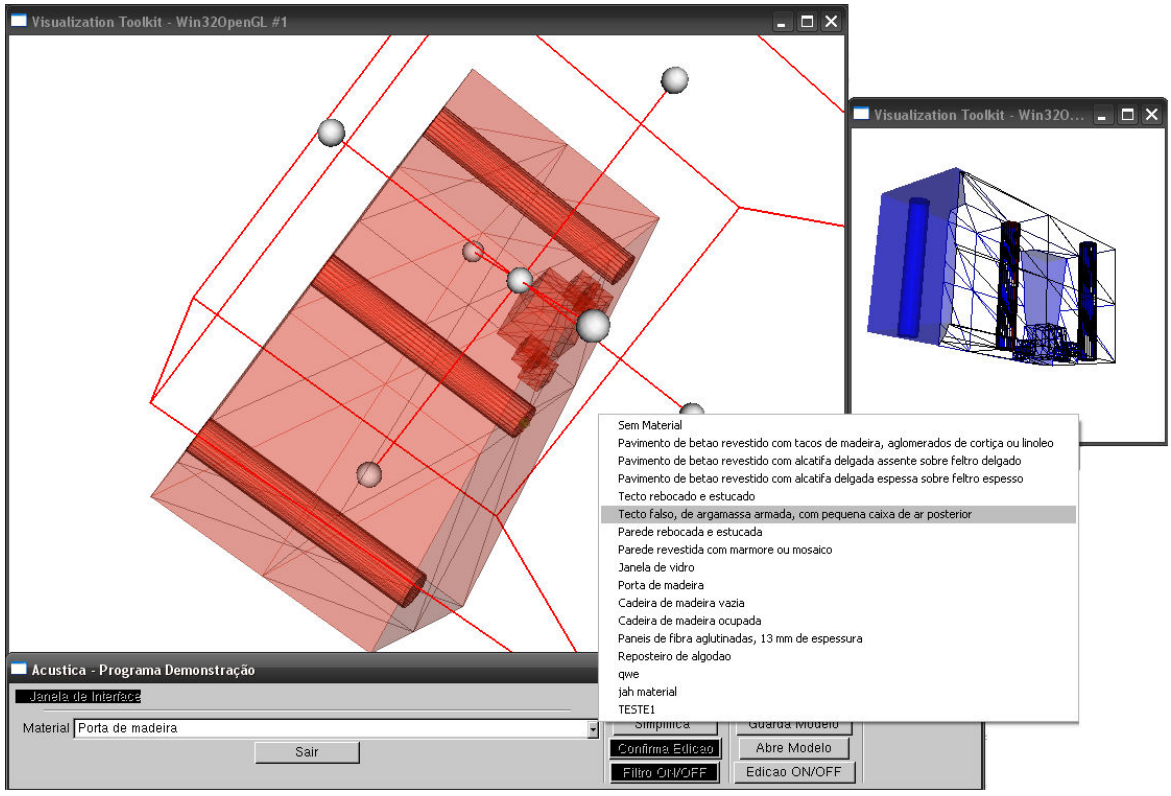


Figura 21 - Interface do programa de correcção acústica de salas

4.2 Reverberação inicial

As primeiras reflexões provêm de posições diferentes do som directo e apresentam tempos de atraso um pouco maiores. Existem diferentes métodos de simulação dessas reflexões. Uma possibilidade é o uso de métodos geométricos, em que as reflexões são calculadas tendo em conta o modelo da sala. Outra é o uso de filtros que tentam recriar a realidade (filtros *tapper delay*) [Park et al, 2006]. Apenas foram estudados os modelos geométricos porque são os que fornecem o melhor compromisso entre precisão e velocidade de cálculo.

4.2.1 Método das imagens virtuais

Existem vários métodos baseados no modelo geométrico propostos na literatura: o método das imagens virtuais [Gardner, 1998] [Allen e Berkley, 1979] [Savioja, 1999], *ray-tracing*[Savioja, 1999][Farina, 1995a], e, mais recentemente, o *cone-tracing* e o *pyramid-tracing* [Farina, 1995a] [Farina, 1995b]. As principais diferenças entre eles consistem na forma como o trajecto das reflexões é determinado. O método das imagens virtuais é um método que consiste em encontrar todos os trajectos possíveis que o som pode efectuar entre a fonte sonora e o ouvinte. O método *ray-tracing* e as suas variantes são métodos estatísticos, que apenas determinam um número específico de raios. Assim, é possível calcular reflexões de ordem elevada cujo custo computacional seja relativamente pequeno, mas sem garantias que todas as reflexões sejam determinadas [Savioja, 1999].

Para este trabalho, optou-se por determinar as primeiras reflexões usando o método das imagens virtuais. O seu princípio básico está ilustrado na Figura 22.

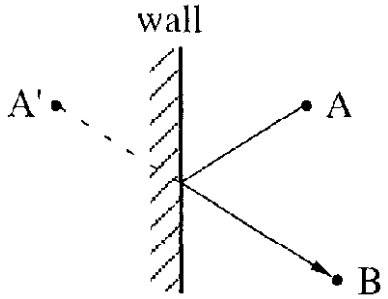


Figura 22 - Método das imagens virtuais (Fonte: [Gardner, 1998])

A fonte encontra-se no ponto A e o utilizador no ponto B. A imagem virtual de A é representada por A'.

O raio emitido pela fonte virtual A' corresponde a uma reflexão de primeira ordem. É possível calcular reflexões de maior ordem através deste método, mas o seu peso computacional aumenta exponencialmente.

4.2.1.1 Algoritmo do método das imagens virtuais

O algoritmo implementado determina as imagens virtuais recorrendo ao modelo 3D da sala. Essa informação é obtida através de um ficheiro no formato OBJ. Este foi o formato escolhido por ser usado no programa de correcção acústica de salas mencionado anteriormente.

Para a implementação do algoritmo foram usadas as funcionalidades fornecidas pela biblioteca gráfica VTK (Ver anexo 1). É relativamente fácil ler um ficheiro *obj* (através do *vtkOBJImporter*) e convertê-lo para um tipo de dados do VTK (*vtkPolyData*). O *vtkPolyData* é um tipo de dados constituído por um número finito de células (uma célula é um polígono convexo definido por n pontos) [Ávila, 2004].

Através da função *IntersectWithLine* é possível determinar se existe intersecção entre dois objectos (um objecto e um segmento de recta). É possível invocá-lo de duas formas distintas:

- Para determinar se um segmento de recta intersecta uma e apenas uma célula especifica do modelo (implementado na classe *vtkCell*).

- Dado um modelo, verificar se existe alguma célula que seja intersectada pelo segmento de recta (implementado na classe *vtkCellLocator*).

A ideia base do algoritmo criado consiste em verificar, com o auxílio da função *IntersectWithLine,* em que células vão ser geradas reflexões entre a fonte sonora e o utilizador. Para isso é necessário determinar, em primeiro lugar, o vector normal da célula, usando para tal o método *vtkPolygon* "*ComputeNormals*". Depois de obtido2 o vector normal é necessário calcular o ponto "espelho" (*mirror point*) da fonte (ou uma posição fantasma calculada anteriormente, se estivermos perante uma reflexão de maior ordem) em relação à célula, que irá ser a sua posição fantasma. Para calcular o ponto espelho precisamos dos seguintes dados: a equação do plano que contem a célula, o vector normal desse plano e a distância da fonte (ou posição fantasma) em relação ao plano.

A equação de um plano pode ser determinada através de 3 pontos que estejam contidos neste, ou através da normal do plano e de um ponto contido do plano.
No nosso caso usou-se o segundo método:

Equação de um plano: $ax + by + cz + d = 0$
em que $d = ax_0 + by_0 + cz_0$

Vector Normal = $\eta = (a, b, c)$
Ponto arbitrario do plano $P = (x_0, y_0, z_0)$

Sabendo que o vector normal do plano é **n = (a,b,c)** e que a distância entre o plano e a fonte define-se por:

$$D = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2 + d^2}} \qquad (4.1)$$

A posição do ponto espelho é então calculada pela seguinte fórmula

$$x_0' = x_0 - 2D\eta \qquad (4.2)$$

$$x_0' = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} - 2\frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2 + d^2}} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \qquad (4.3)$$

Na Figura 23 encontra-se descrito o algoritmo usado para calcular as primeiras reflexões de cada fonte sonora.

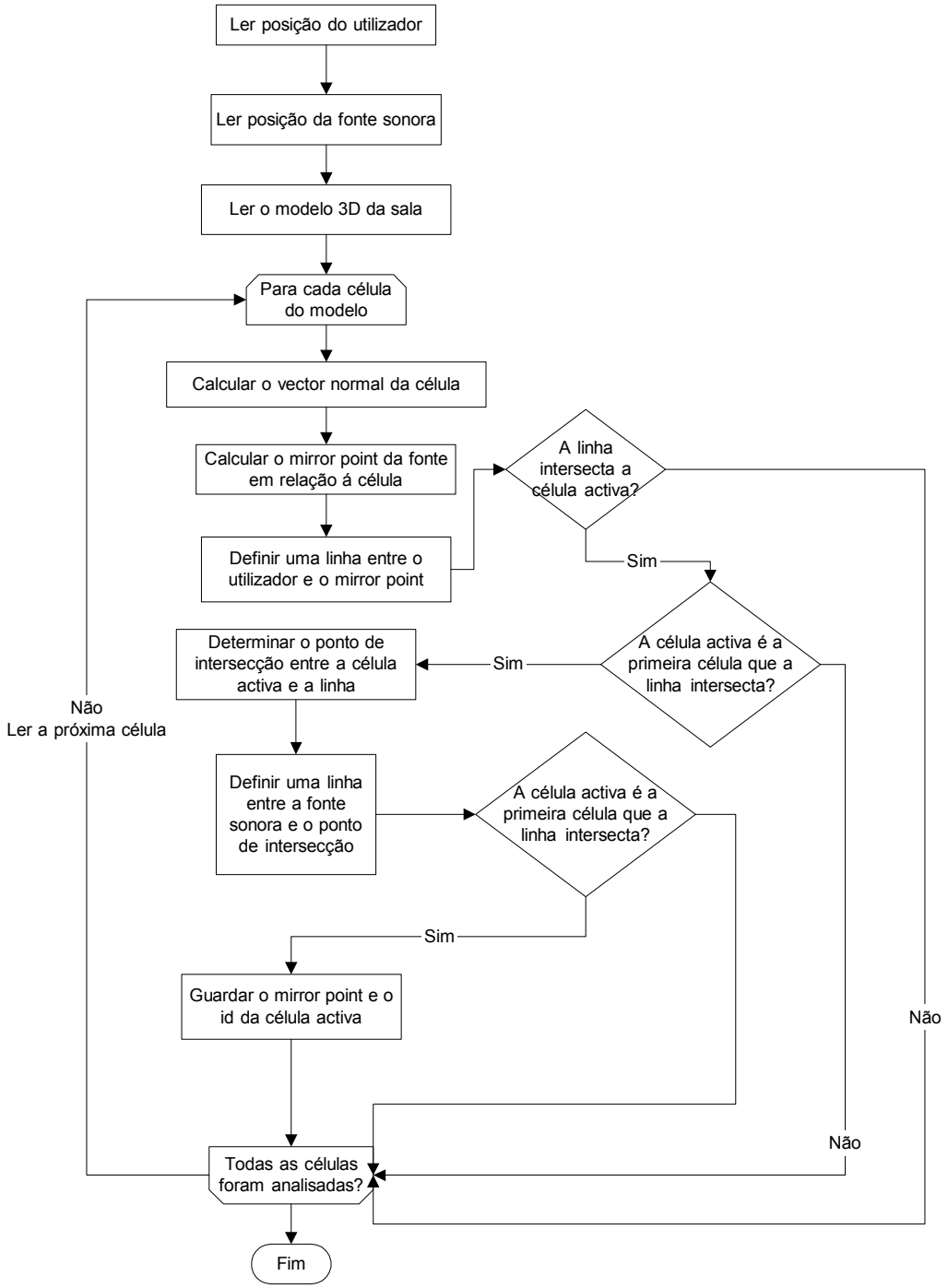


Figura 23 - Algoritmo das imagens virtuais

4.2.1.2 Resultados

As figuras seguintes ilustram os resultados obtidos pelo algoritmo implementado.

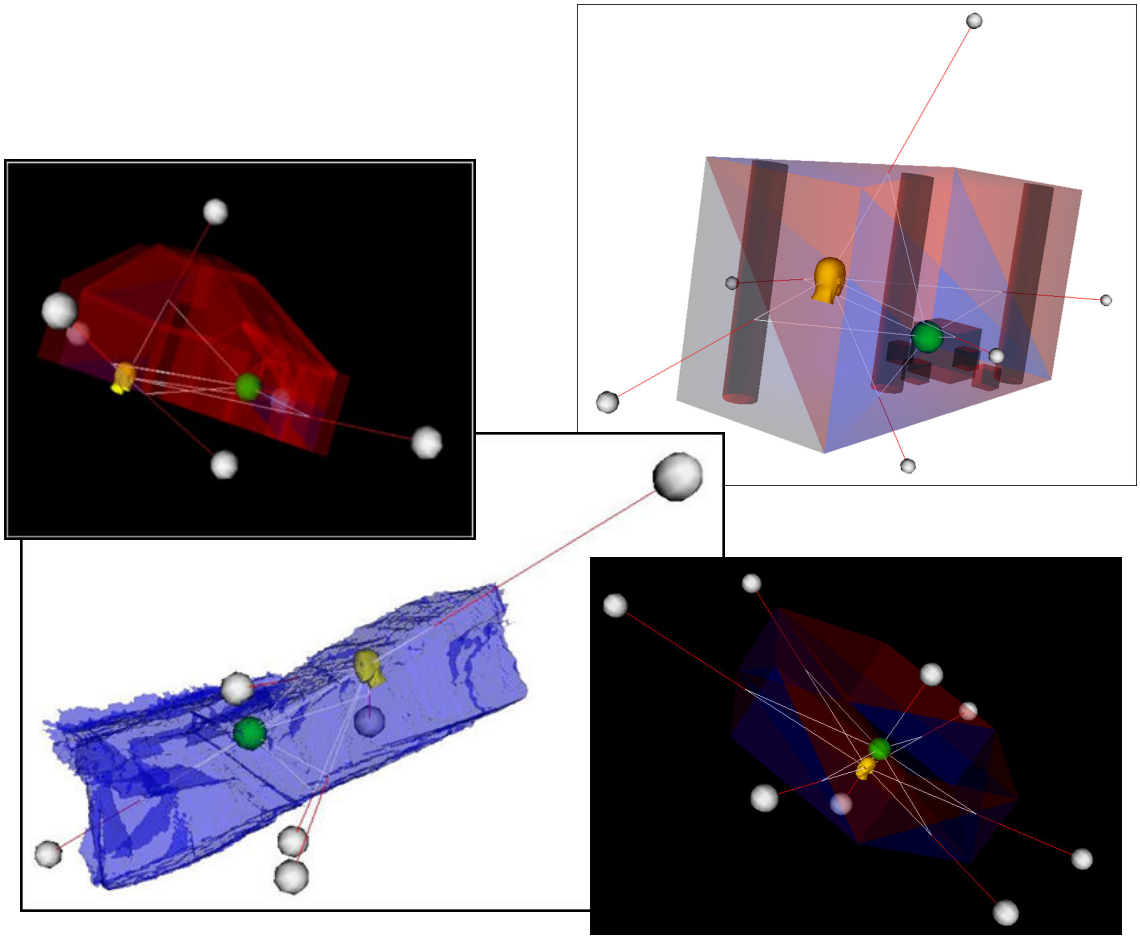


Figura 24 - Exemplo das primeiras reflexões calculadas com o método das imagens virtuais

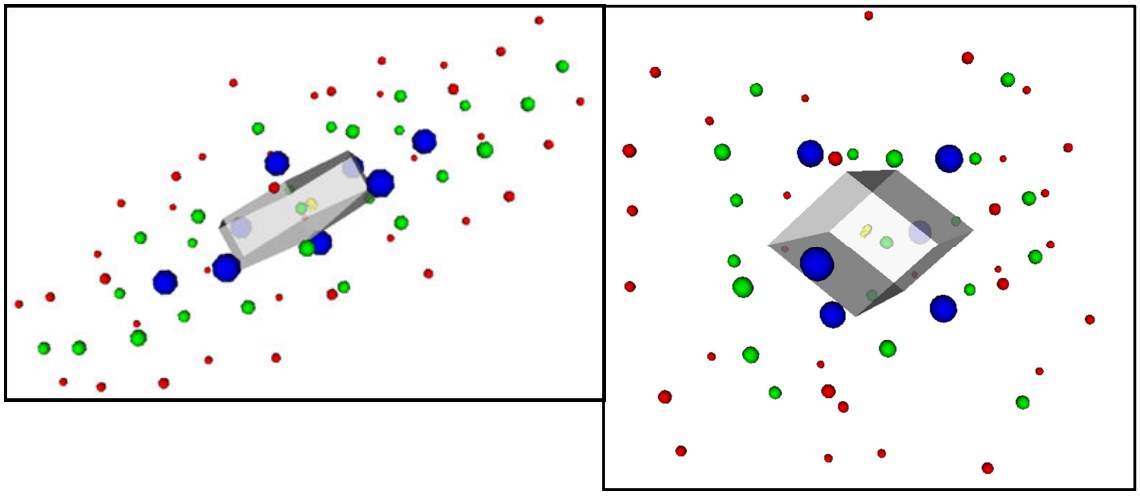


Figura 25 - Exemplo de reflexões de 3ª ordem calculadas com o método das imagens virtuais

4.2.2 Primeiras reflexões

Aplicando este método é possível considerar as reflexões como fontes distintas. A única diferença reside no facto dos sinais de saída dessas fontes serem atenuados tendo em conta o coeficiente de absorção dos materiais onde ocorrerem reflexões. Como cada material tem coeficientes de absorção distintos para cada banda, é necessário filtrar o sinal de entrada em bandas específicas, tratando cada uma separadamente. Dessa forma cria-se o som que irá ser reproduzido em cada reflexão que foi obtida.
O algoritmo encontra-se no fluxograma seguinte:

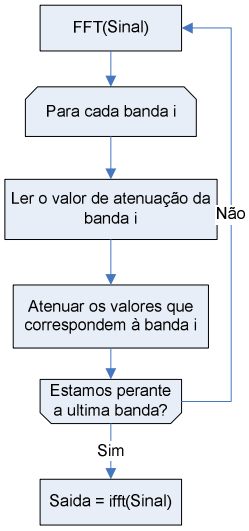


Figura 26 - Fluxograma do algoritmo para atenuação do som por bandas

Como os valores não estão a ser atenuados no domínio do tempo, a frequência intermédia entre duas bandas não é obtida de forma linear mas sim tendo por base a escala logarítmica. A fórmula usada para determinar esse valor é:

$$frequência_{limiar} = \sqrt{frequência_{banda1} * frequência_{banda2}} \quad (4.4)$$

Por exemplo, o valor limiar entre as bandas centradas em 250 Hz e 500 Hz não é 375 Hz mas sim 354 Hz. Na Figura 27 encontra-se um exemplo da forma como o som é atenuado. No primeiro gráfico mostra-se um sinal com espectro de frequência plana. No segundo gráfico encontra-se o espectro desse sinal já atenuado, tendo em conta os valores de atenuação descritos na legenda da figura.

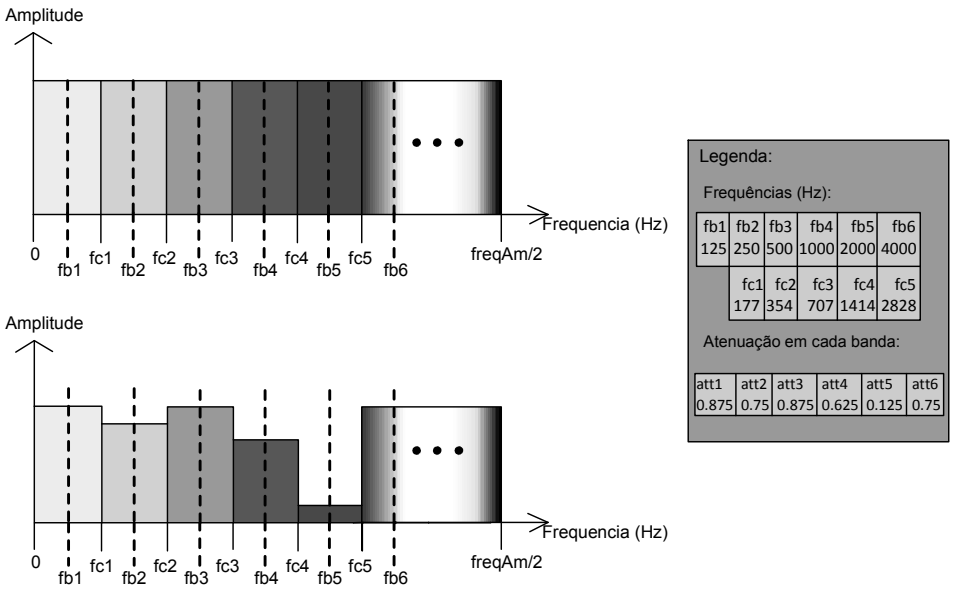


Figura 27 - Exemplo da atenuação por bandas de frequência

As imagens seguintes mostram alguns exemplos obtidos pelo algoritmo de atenuação por bandas descrito no fluxograma anterior. Apenas foram considerados reflexões de primeira ordem.

O som de entrada do algoritmo foi ruído branco gaussiano - um sinal que contém todas as frequências. Na Figura 28 vemos o seu espectro de frequência.

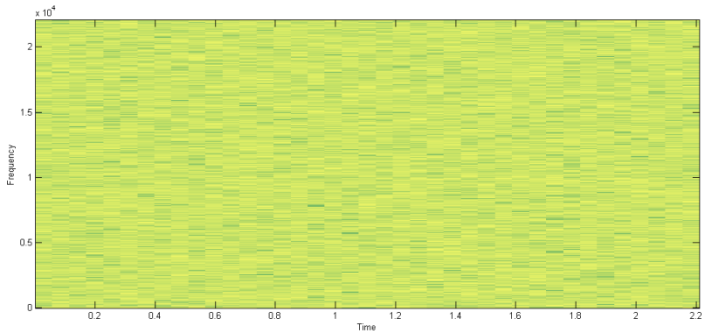


Figura 28 - Espectro inicial do sinal

Podemos observar que o espectro se encontra todo com a mesma cor, implicando assim que todas as frequências estão presentes de uma forma uniforme. Agora, através do programa de correcção acústica de salas gerou-se uma sala em que todos os materiais pertencentes à mesma tinham os coeficientes de absorção: [1, 1, 1, 1, 0, 1] para as bandas centradas em [125, 250, 500, 1000, 2000, 4000] respectivamente, ou seja apenas a banda dos 2000 Hz não vai absorver as frequências. O espectro de frequências do sinal obtido está descrito na Figura 29. É possível verificar que o sinal apenas contém uma banda de frequência, sendo as restantes eliminadas.

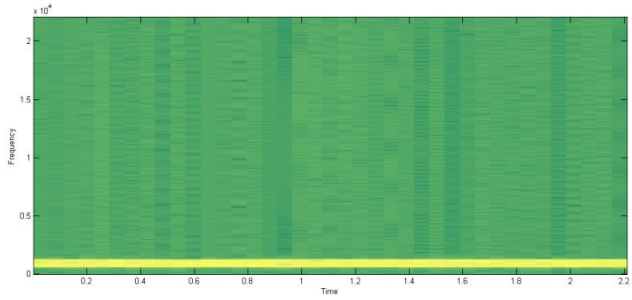


Figura 29 - Exemplo da atenuação efectuada no ruído branco

Na Figura 30 podemos ver outro exemplo em que os coeficientes dos materiais da sala são [0,1,0,1,1,0].

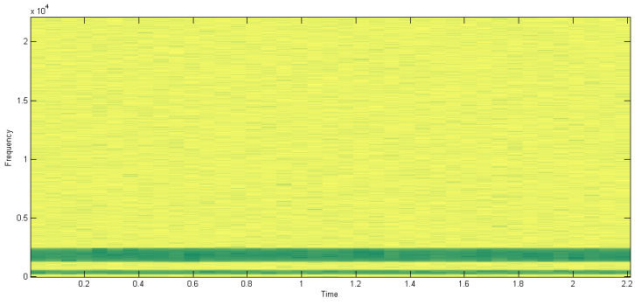


Figura 30 - Exemplo da atenuação efectuada

4.3 Reverberação tardia

A reverberação difusa ou tardia é um tópico estudado há vários anos existindo por isso uma extensa literatura e múltiplas unidades de reverberação diferentes. Os primeiros passos foram dados por *Schroeder* em 1962 com um artigo no qual são descritos os parâmetros necessários para criar reverberação incolor, ou seja, com uma resposta em frequência uniforme para todas as frequências [Moorer, 1987]. Este artigo descreve algumas das propriedades mais importantes das salas de espectáculo de grande dimensão, tais como a resposta em frequência e a forma como um som impulsional é percepcionado. Encontra-se ainda descrito o filtro *all-pass*, os motivos que levaram à sua implementação e uma unidade de reverberação baseada em filtros *all-pass*. Estes filtros encontram-se explicados em maior detalhe no ponto 4.3.1.

Segundo *Schroeder,* para se conseguir recriar a reverberação difusa é necessário simular um mínimo de 1000 reflexões. Tentar simular essas reflexões com o método das imagens virtuais é impossível, devido ao peso computacional envolvido. É então necessário criar algoritmos com um menor custo computacional.

As condições necessárias a atingir em reverberação artificial são0:

1- A resposta em frequência tem que ser uniforme para todas as bandas de frequência.
2- O reverberador tem que cobrir a gama de frequência existentes no som a ser reverberado.
3- A quantidade de ecos após curto intervalo de tempo desde o inicio da reprodução do som tem que ser suficientemente grande para que os sons não sejam perceptíveis individualmente pelo ouvido.
4- Não devem existir 'ecos' periódicos
5- A resposta amplitude - frequência não pode conter nenhuma periodicidade óbvia.

Moorer, em 1987, resume e explica as primeiras unidades de reverberação descobertas de *Schroeder* tais como o filtro comb e all-pass, e sugere combinações criadas com esses filtros para melhorar a resposta audível, tais como os filtros *comb* e *all-pass* oscilatórios, discutindo as vantagens e desvantagens do seu uso. A optimização que *Moorer* propõe a uma das combinações de filtros de *Schroeder* ficou conhecida como reverberador de *Moorer*. Os parâmetros necessários para a sua implementação, tais como o atraso de cada filtro usado na unidade de reverberação e o uso de filtros passa baixo para simular a atenuação do ar, são discutidos no mesmo artigo. Outros dos pontos que foi inicialmente descrito neste artigo e que ainda hoje continua em estudo é a possibilidade da criação de resposta impulsionais sintéticas de salas recorrendo para isso a ruído branco (tendo em conta que a resposta impulsional de uma sala se assemelha a ruído branco com um decaimento exponencial) [Moorer, 1987].

Em [Farina, 1993], *Angelo Farina* descreve como gerar reverberação por convolução de respostas impulsionais previamente gravadas com sons anecóicos, apresentando resultados sobre o peso computacional que esta operação exige para diferentes sistemas.

No livro *Applications of Digital Signal Processing to Audio and acoustics* [Gardner, 1998] existe um capitulo que descreve com bastante detalhe a evolução dos algoritmos de reverberação existentes desde os filtros *comb* passando pelas *Feedback Delay Networks* (FDN) e *Digital Waveguide Mesh* (DWM), até aos algoritmos com variações temporais, explicando detalhadamente os passos necessários para a sua implementação.

Mais recente, o livro *Digital Audio Effects*[Zölzer, 2002] fornece também alguns algoritmos de reverberação dando maior ênfase à FDN e apresentando mais detalhes técnicos sobre a sua implementação. Descreve de uma forma mais científica alguns dos valores a usar na FDN tais como os atrasos das linhas da FDN.

Um dos principais objectivos para o uso das FDN consiste no aumento da densidade de ecos simulados. Em 1997, *Davide Rocchesso* descreveu uma matriz de realimentação para FDN baseada na sequência de *Galois* e ordenada numa matriz circular de forma a maximizar a densidade de ecos. Devido à particularidade dessa sequência e da matriz circular é possível optimizar o peso computacional das FDN [Rocchesso, 1997].

Em 2000, *Luke Dahl* e *Jean Marc Jot* descrevem um filtro baseado numa nova implementação do filtro *all-pass*. Esta implementação tem como objectivo aumentar a precisão do tempo de decaimento e o nível de energia dos filtros *all-pass*. Descrevem ainda uma unidade de reverberação baseada numa *Unitary* FDN (UFDN) que recorre a esses filtros [Dahl et al, 2000].

Em 2006 foi descrito um algoritmo de reverberação artificial baseado num *phase vocoder* 0. Embora a resposta não seja melhor que os algoritmos inventados até à data, tem a vantagem de ser mais rápido.

Nos pontos seguintes vão ser explicados algumas das unidades de reverberação estudadas e implementadas. Todas as unidades de reverberação aqui descritas foram implementadas cronologicamente, desde as mais básicas (filtros *Comb*) até às unidades *Multirate*. O objectivo deste estudo passa por determinar um algoritmo que simule a reverberação tardia real. Algumas das unidades de reverberação foram implementadas apenas para ter uma noção do resultado produzido, conhecendo-se à partida as suas vantagens e defeitos. No entanto, optou-se por esta estratégia para assim se ter uma relação qualidade/custo de vários algoritmos e verificar qual o mais indicado para cada ambiente virtual criado.

4.3.1 Reverberador de *Schroeder*

Na Figura 31 encontra-se descrito um filtro *comb*. Este filtro consiste numa linha de atraso cuja saída é recirculada para a entrada. A função de transferência para este filtro é:

$$H(z) = \frac{z^{-m}}{1-gz^{-m}} \quad (4.5)$$

onde *m* é o atraso aplicado ao sinal e *g* o valor de atenuação. Para este filtro ser estável é necessário que *g* <1. A Figura 31 mostra a resposta impulsional do filtro.

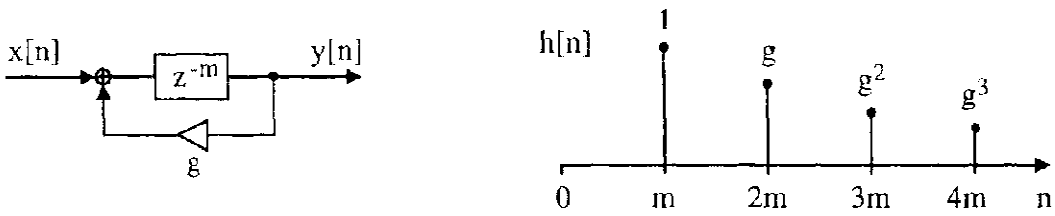


Figura 31 - Resposta impulsional e desenho do Filtro comb (Fonte: [Gardner, 1998])

Schroeder determinou que um filtro *comb* podia ser modificado facilmente para fornecer uma resposta em frequência plana misturando o sinal de entrada e a saída do filtro *comb*, como é possível observar na Figura 32. O filtro resultante dessa modificação é denominado de *all-pass*. A sua resposta em frequência é:

$$H(z) = \frac{z^{-m}-g}{1-gz^{-m}} \quad (4.6)$$

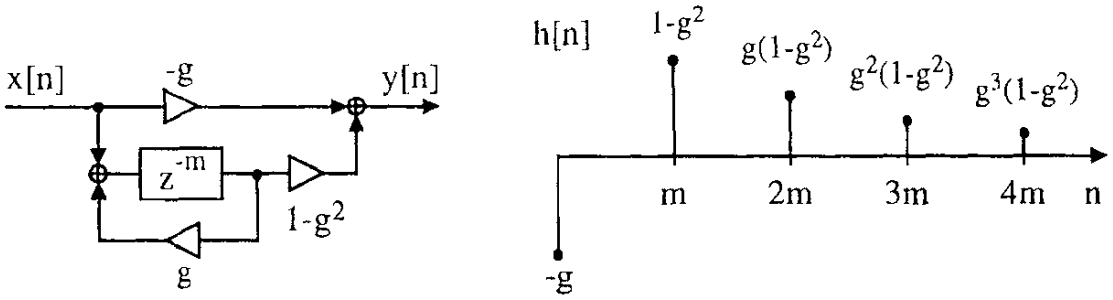


Figura 32 - Resposta impulsional e desenho de um filtro all-pass (Fonte: [Gardner, 1998])

Embora o filtro *all-pass* forneça uma magnitude igual para todas as frequências, o resultado obtido é parecido com o do filtro *comb,* ou seja, continua a ser criada uma certa coloração no timbre do resultado de saída [Gardner, 1998].

É possível criar uma unidade de reverberação apenas com um filtro *all-pass* ou um filtro *comb*. No entanto, para simular a cauda de reverberação é necessário simular um mínimo de 1000 reflexões do som, em que não seja perceptível periodicidade. Logo, simular a reverberação apenas com filtros *comb* ou *all-pass* torna-se impraticável pois as suas respostas não conseguem simular essa densidade de reflexões. O resultado final iria ser próximo de ecos espaçados em intervalos de tempo uniformes e a qualidade do som seria extremamente artificial, com um certo tom metálico.

Uma das optimizações que Schroeder efectuou para melhorar o resultado da reverberação consistiu em ligar vários filtros *comb* em paralelo seguido de dois filtros *all-pass* em série (Figura 33).

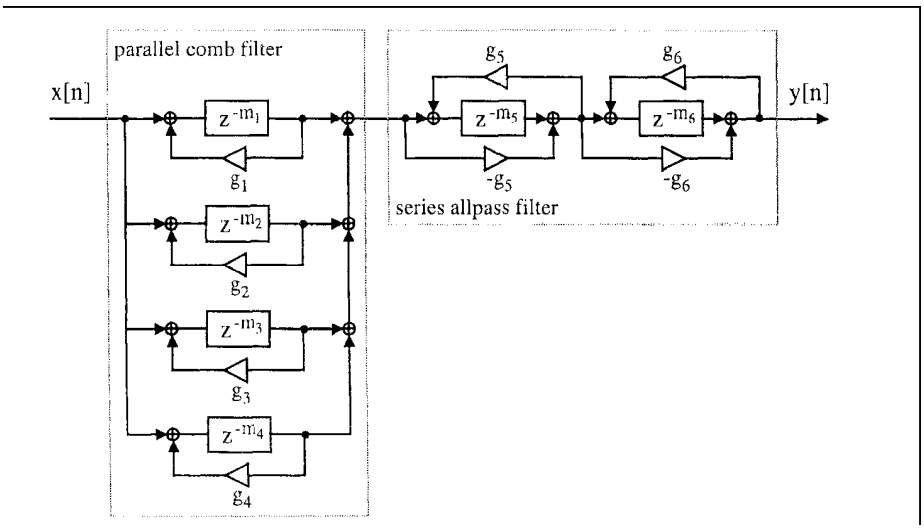


Figura 33 - Filtros comb em paralelo (Fonte: [Gardner, 1998])

O tempo de decaimento dos filtros é definido pelo valor de atenuação g_i. Através da fórmula seguinte é possível calcular o decaimento de cada filtro tendo em conta tempo de reverberação específico:

$$g_i = 10^{-3m_i T/T_r} \qquad (4.7)$$

onde m_i é o número de amostras de atraso do filtro, T o período de amostragem e T_r o tempo de reverberação.

Os atrasos dos filtros são escolhidos de forma a ter uma razão de 1.5 entre o maior e o mais pequeno (*Shroeder* sugeriu uma diferença entre 30 a 45 ms). Através do paralelismo usado nos filtros *comb* produz-se o tempo de reverberação pretendido. Os filtros *all-pass* foram inseridos apenas para aumentar a densidade de ecos simulados pelos filtros *comb*. *Schroeder* sugeriu os seguintes parâmetros [Moorer, 1987]:

-Atrasos:5 e 1.7*ms*

-Ganhos: 0.7

De forma a minimizar os batimentos que podem existir é sugerido que o número de amostras correspondente aos atrasos de cada filtro *comb* seja um número primo. O valor recomendado para o atraso em cada filtro está entre 50 a 100 ms.

4.3.1.1 Densidade de ecos e densidade modal

Dois critérios a ter em conta para se obter um maior realismo nos algoritmos de reverberação são a densidade modal e a densidade de reflexões. A densidade modal consiste nos modos de ressonância da sala e a densidade de ecos o número de reflexões simuladas. A densidade modal dos filtros *comb* paralelos expressa em número de modos por Hz é:

$$D_m = \sum_{i=0}^{N-1} \tau_i = N.\tau \quad (4.8)$$

Onde τ_i é o atraso i em segundos, e τ o atraso médio. O atraso total dos filtros *comb*, expresso em s é igual à densidade modal dos filtros *comb* paralelos expresso em número de modos por Hz. Ao relacionar isto com a densidade máxima das salas reais, obtêm-se a seguinte relação entre o tempo total dos atrasos e o maior tempo de reverberação a simular [Gardner, 1998]:

$$\sum_i \tau_i = D_m > D_f \approx \frac{T_{max}}{4} \quad (4.9)$$

onde D_f é a densidade da frequência máxima tendo em conta o modelo estatístico da reverberação tardia e T_{max} é o tempo de reverberação máximo pretendido.

Na prática, uma baixa densidade modal leva ao aparecimento de batimentos audíveis, resultado dos sinais com bandas estreitas. Um sinal com bandas estreitas pode excitar dois modos vizinhos que irão vibrar por simpatia à sua equação diferencial. Para atenuar este problema o espaçamento médio dos modos pode ser escolhido de forma que o período médio de cada batimento seja pelo menos igual ao tempo de reverberação, levando assim à seguinte relação:

$$\sum \tau_i \geq T_{max} \quad (4.10)$$

A densidade de ecos dos filtros *comb* é a soma da densidade de ecos dos filtros *comb* individuais. Cada filtro *comb* calcula uma saída por cada τ_i; assim, a densidade de ecos combinada , expressa pelo número de ecos por segundo é:

$$D_e = \sum_{i=0}^{N-1} \frac{1}{\tau_i} \approx \frac{N}{\tau} \quad (4.11)$$

Esta aproximação apenas é válida se os atrasos forem muito próximos.

4.3.1.2 Produzir saídas descorrelacionadas

O reverberador descrito no ponto anterior é um reverberador monofónico com uma entrada e uma saída. Schroeder sugeriu um método para produzir múltiplas saídas através de combinações lineares das saídas dos filtros comb. Para isso os filtros *all-pass* eram inseridos antes dos filtros comb como ilustra a figura 34.

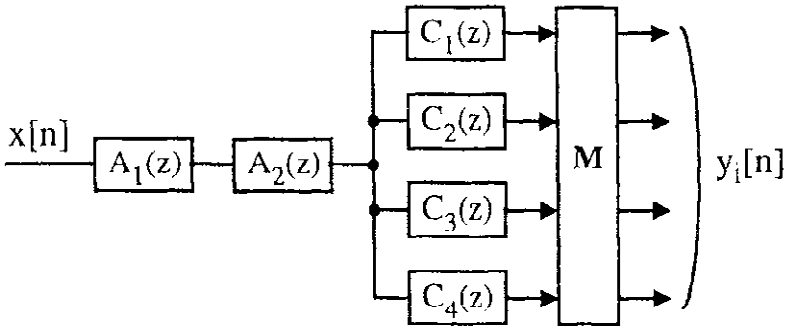


Figura 34 - Matriz para produzir saídas descorrelacionadas (Fonte: [Gardner, 1998])

De seguida era usada uma matriz para misturar as saídas dos filtros *comb* em que o número de linhas era igual ao número de filtros. Schroeder sugeriu que os coeficientes da matriz apenas contivessem valores de 1 ou -1 e Jot sugeriu o uso de colunas ortogonais. O objectivo desta matriz consistia em criar saídas que fossem mutuamente descorrelacionadas. Por exemplo, a matriz descrita no ponto seguinte produz uma reverberação estéreo bastante envolvente quando reproduzida em auscultadores.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$$

É possível misturar duas saídas que sejam mutuamente descorrelacionadas e obter o valor pretendido para a descorrelação cruzada através das fórmulas seguintes (dando como exemplo um reverberador com duas linhas de saída):

$$y_L(t) = \cos(\theta)\, y_1(t) + \sin(\theta)\, y_2(t)$$

$$y_R(t) = \sin(\theta)\, y_1(t) + \cos(\theta)\, y_2(t) \qquad (4.13)$$

$$\theta = \arcsin(\mathrm{IACC})\,/2$$

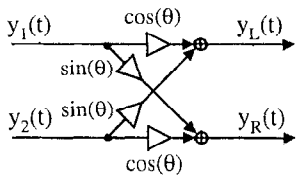


Figura 35 - Controlo da descorrelação cruzada na reverberação binaural

O resultado auditivo final com este algoritmo melhora consideravelmente, quando comparado com os filtros *comb* e *all-pass* individualmente, desaparecendo a periodicidade de ecos existente.

Um dos primeiros requisitos que um algoritmo de reverberação tem que cumprir para simular a realidade consiste em fornecer uma resposta com um RT60 (tempo de reverberação) específico. Este algoritmo cumpre esse requisito, no entanto, para tempos de reverberação elevados o som continua a ser extremamente artificial. Algumas das falhas descritas por este algoritmo são [Gardner, 1998]:

- A resposta inicial soa de uma forma muito discreta, fazendo que a qualidade do som fique um pouco granulada, faltando uma certa fluidez, principalmente para sons muito curtos e impulsivos.
- A amplitude da resposta tardia, ao contrário de ter um decaimento suave, exibia uma modulação pouco natural, resultando numa certa agitação ou criando batimentos no som.
- Para tempos de reverberação grandes, o resultado audível do algoritmo não é natural, sendo adicionado um certo timbre metálico.

Devido à má qualidade do resultado final deste algoritmo optou-se por estudar o algoritmo de *Moorer* (ponto seguinte).

4.3.2 Reverberador de *Moorer*

O algoritmo de *Moorer* foi criado tendo como base o algoritmo de *Schroeder* com o objectivo de suprimir os problemas do algoritmo anterior, ou seja, eliminar o som metálico, diminuir os batimentos e a granularidade do resultado de saída. A primeira alteração consistiu em aumentar o número de filtros *comb* para 6, aumentando assim a densidade de ecos simulados e melhorando os resultados para tempos de reverberação elevados. Outra das alterações consistiu em inserir um filtro *passa baixo* em cada filtro *comb*.

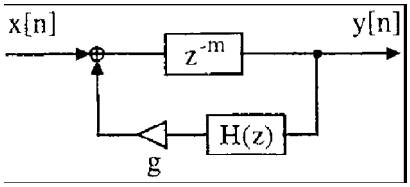


Figura 36 - Filtro Comb com o filtro passa baixo (Fonte: [Gardner, 1998])

As frequências de corte usadas no algoritmo tentavam simular a absorção do som provocada pelo ar e eram determinadas através de medições reais. Através dos valores medidos da absorção do ar verificou-se que as frequências mais elevadas do som tinham um decaimento maior que as frequências mais baixas. Assim, ao serem inseridos filtros passa baixo em cada filtro, as frequências mais altas estão a ser atenuadas mais rapidamente, atenuando o realismo do som de saída do algoritmo. Outro benefício obtido com estas alterações foi a diminuição do efeito de granularidade em sons impulsivos. No entanto, o resultado final ainda está um pouco distante da reverberação real.

Moorer criou uma tabela com alguns valores que ganhos que devem ser usados, no entanto, os valores são muito gerais. Para valores intermédios é possível calcular a interpolação para o valor pretendido. Os valores de atraso e de ganho dos filtros *all-pass* continuam a ser definidos através da proposta de *Schroeder*.

O resultado final deste algoritmo já é substancialmente melhor, sendo uma boa opção para tempos de reverberação pequenos. No entanto, para tempos de reverberação mais longos ainda existe alguma coloração tonal no som, dando a impressão que o som está a vir de um tubo metálico. A granularidade para sons impulsivos é bastante menor mas ainda perceptível. Este algoritmo é uma boa escolha para gerar um efeito de reverberação simples. No entanto, devido aos problemas apontados anteriormente não é o algoritmo indicado para simular a reverberação real tardia.

4.3.3 Resposta Impulsional Sintética

Outro algoritmo para gerar reverberação consiste em gerar artificialmente a resposta impulsional de uma sala para posterior convolução com o som de entrada. Essa resposta pode ser gerada partindo do principio que as respostas impulsionais reais são bastante parecidas a ruído branco com um decaimento exponencial [Moorer, 1987]. Para criar uma resposta impulsional sintética é necessário gerar um sinal de ruído branco gaussiano com uma dada duração e com o tempo de reverberação pretendido. De seguida é atenuado o sinal para ter um decaimento exponencial dependente do tempo de decaimento específico. Isso é obtido através da seguinte fórmula:

$$S_{IR(t)} = WN(t)e^{(-6\times t\times \log 2)/_{Tr}\times freqAm} \quad (4.14)$$

Em que *WN(t)* é o ruído branco gerado, *Tr* o tempo de reverberação pretendido e *freqAm* a frequência de amostragem. Caso seja necessário simular tempos de reverberação dependentes da frequência é possível filtrar previamente o ruído branco nas bandas de frequência pretendida. Assim, é possível aplicar atenuações distintas em cada banda. De seguida, somam-se os sinais filtrados já atenuados obtendo assim a resposta impulsional sintética.

Por fim, após ter sido gerada a resposta impulsional, apenas é necessário calcular a convolução com o som de entrada pretendido e obtemos o som reverberante. Se o objectivo do uso da resposta impulsional sintética consiste em simular a reverberação tardia então é necessário efectuar alguns cálculos adicionais, pois a resposta deste método não define correctamente o tempo em que é iniciada a cauda de reverberação, calculando a resposta logo após a activação do som inicial. A Figura 37 mostra um pequeno exemplo dos cálculos adicionais que necessitam de ser efectuados para combinar as primeiras reflexões com a reverberação tardia. Assumindo que o som de entrada é um som impulsivo então, em a), é possível visualizar o início do som directo que chega ao ouvinte. Em b) podemos ver o resultado da convolução do som directo com a resposta impulsional sintética. No entanto, como o objectivo passa apenas por simular a cauda da reverberação e assumindo que as primeiras reflexões são entre as amostras x_o e x_i os valores calculados na convolução com esses pontos são desprezáveis. Para evitar que a passagem entre as primeiras reflexões e a reflexão tardia seja perceptível é necessário interpolar o resultado final a partir de um certo ponto de forma a essa passagem ser efectuada de forma suave (como se encontra exemplificado em c)).

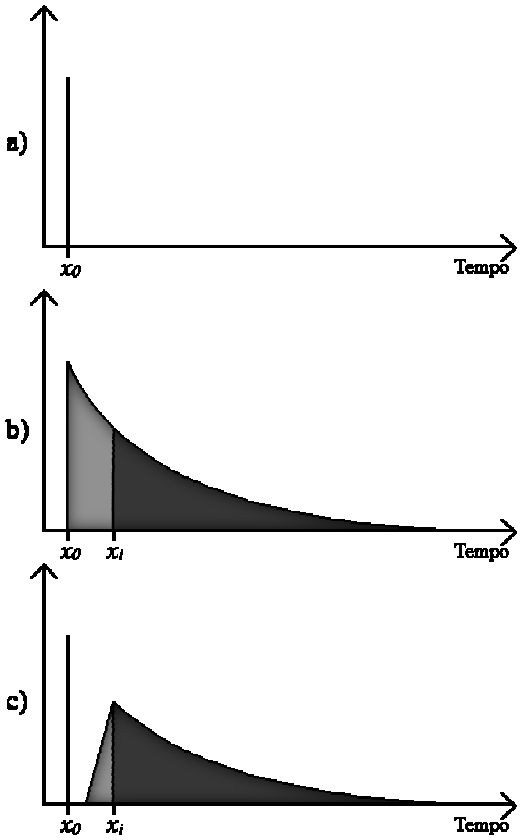


Figura 37 - Atenuação a efectuar à resposta impulsional sintética

Os resultados obtidos através deste método são satisfatórios. O som resultante deste método é bastante natural. Comparando com os algoritmos anteriores comprova-se que o resultado é mais natural. No entanto, como este é um método com um peso computacional elevado e com pouca flexibilidade para alterar os seus parâmetros não é possível usá-lo em ambientes que necessitem de gerar a reverberação em tempo real.

4.3.4 *Feedback Delay Networks* (FDN)

Como foi descrito anteriormente, usando apenas filtros *comb* individualmente o resultado som de saída tem fraca qualidade; no entanto se aumentarmos o número de filtros usados a qualidade vai melhorando substancialmente. Em 1982 *Stautner* e *Puckette* apresentaram uma estrutura para gerar reverberação artificial baseada em linhas de atraso interligadas num *anel* de realimentação através de uma matriz. As estruturas desse tipo começaram a ser denominadas por *Feedback Delay Networks* (FDN). A unidade de FDN proposta por ambos foi obtida como uma generalização do vector do filtro comb recursivo

$$y(n) = x(n-m) + g\,.y(n-m),$$

Onde a linha de atraso de *m-amostras* foi substituída por um grupo de linhas de atraso com diferentes tamanhos e o ganho de realimentação substituído pela matriz de realimentação0. Assim, a matriz de realimentação permite recircular a saída de cada linha de atraso para diferentes entradas. A estrutura comporta-se como o algoritmo de Schroeder quando a matriz de realimentação é diagonal. Permite uma densidade muito maior de que os filtros 'comb' em paralelo. Os atrasos de cada linha são escolhidos tendo em conta as sugestões de Schroeder (ponto 4.3.1).

É possível garantir estabilidade da FDN se a matriz de realimentação for escolhida de forma a ser o produto de uma matriz unitária com um ganho *g*, onde *g* <1. A matriz sugerida por *Stautner* e *Puckette* é:

$$\mathrm{A} = \frac{\mathrm{g}}{\sqrt{2}} \begin{bmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \quad (4.15)$$

Em que g controla o tempo de reverberação

- As saídas são mutuamente incoerentes, podendo ser usadas em sistemas de som 4.1 (sistemas de reprodução com quatro colunas e um *subwoofer*), para assim simular a reflexão difusa
- As perdas por absorção do ar podem ser simuladas inserindo um filtro passa baixo em cada linha de atraso (como no algoritmo de *Moorer*).

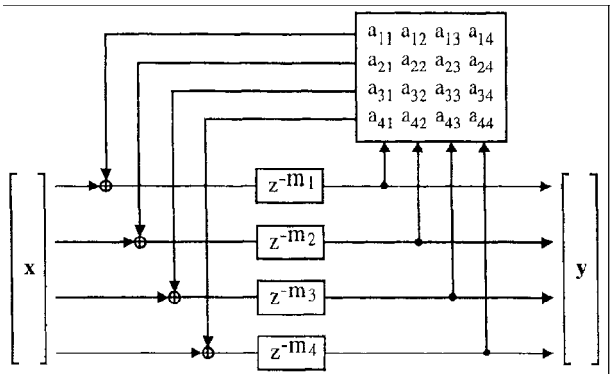


Figura 38 - *Feedback Delay Network* proposto por *Stautner* e *Puckette* (Fonte: [Gardner, 1998])

Este algoritmo cria alguma agitação e uma coloração tonal no som, que se encontra presente na reverberação tardia. Os autores atribuíram essa agitação aos batimentos dos modos adjacentes. Para reduzir o espaçamento é usada a seguinte fórmula:

$$\sum_i \tau_i \geq T_{max} \quad (4.16)$$

onde τ_i é o atraso em cada linha de atraso e ***Tmax*** o tempo de reverberação. Para reduzir a coloração tonal, é sugerido variar o tamanho dos atrasos de uma forma aleatória [Gardner, 1998].

A unidade de reverberação descrita na Figura 38 é a base de algumas novas unidades de reverberação. *Jean Marc Jot* foi um dos criadores de uma dessas novas unidades, criada tendo em conta duas propriedades. Uma unidade de reverberação pode ser desenhada:

- com um tempo e uma densidade de frequências arbitrários garantindo simultaneamente a ausência de coloração tonal na reverberação tardia
- tendo em conta vários tempos de reverberação $T_r(w)$ e a envolvência da resposta em frequência $G(w)$.

Para isso parte-se de um sistema conservador de energia, denominado por protótipo *lossless* (como o ilustrado na figura 38) cuja resposta impulsional é perceptualmente igual a ruído branco. Para se obter um tempo de reverberação dependente da frequência são associados filtros de absorção a cada linha de atraso do sistema. O método inicial proposto por Jot para inserir essas perdas de absorção passou por inserir um ganho g_i em cada linha de atraso. O logaritmo do ganho é proporcional ao tamanho do atraso.

$$g_i = a^{m_i} \quad (4.17)$$

Dessa forma, o tempo de reverberação é dado por [Zölzer, 2002]:

$$T_r = \frac{-3T_s}{\log(a)} \quad (4.18)$$

Logo, das duas fórmulas anteriores deduz-se que a é

$$a = 10^{(-3.T_s/T_r)} \quad (4.19)$$

Nesse caso, para uma *FDN* com 3 linhas de atraso os cálculos de saída seriam calculados da seguinte forma:

- Sendo a nossa matriz de realimentação unitária

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}, \quad (4.20)$$

os coeficientes de atenuação $\mathbf{g_i}$ e os atrasos em cada linha definidos por $\mathbf{M_i}$.

- Os cálculos efectuados na FDN que geram a realimentação interna da unidade são:

$$\begin{bmatrix} x_1(n) \\ x_2(n) \\ x_3(n) \end{bmatrix} = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & g_3 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x_1(n-M_1) \\ x_2(n-M_2) \\ x_3(n-M_3) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} u(n) \quad (4.21)$$

E a saída final é definida por:

$$v(n) = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x_1(n-M_1) \\ x_2(n-M_2) \\ x_3(n-M_3) \end{bmatrix} \quad (4.22)$$

De forma a atenuar e obter um decaimento maior nas altas frequências, é possível inserir um filtro passa baixo em cada linha de atraso. Dessa forma é possível fazer com que o tempo de decaimento de cada linha de atraso $T_r(w)$ seja dependente da frequência. O filtro é escolhido para que o logaritmo da sua resposta em magnitude seja proporcional ao tempo de reverberação:

$$20log_{10}|h_i(e^{jw})| = \frac{-60T}{T_r(w)} m_i \quad (4.23)$$

No entanto, inserir os filtros passa baixo na unidade *lossless* faz com que a resposta em frequência deixe de ser uniforme para todas as frequências. Para corrigir esta falha é possível associar um filtro de correcção t(z) em série com o filtro de referência, cujo quadrado da magnitude é inversamente proporcional ao tempo de reverberação:

$$|h_i(e^{jw})| \propto \frac{1}{\sqrt{T_r(w)}} \quad (4.24)$$

Após inserir este filtro obtém-se uma resposta em frequência uniforme. A unidade final, resultante destas alterações encontra-se descrita na Figura 39:

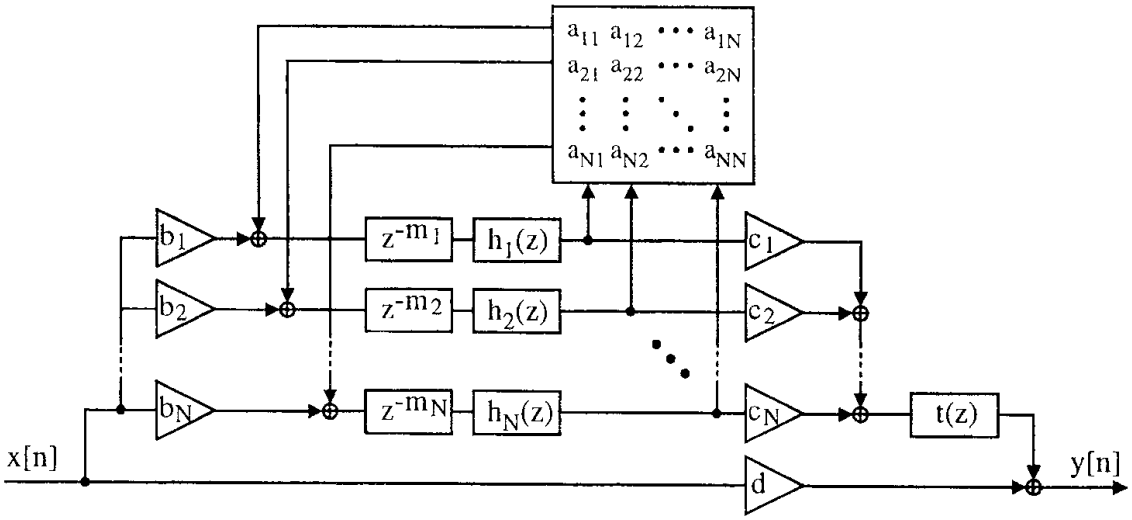


Figura 39 – FDN com os filtros de absorção e correcção (Fonte: [Gardner, 1998])

O filtro de absorção proposto por Jot é um filtro IIR e tem a seguinte função de transferência:

$$H_i(z) = k_i \frac{1-\beta_i}{1-\beta_i z^{-1}} \quad (4.25)$$

Cuja equação diferencial é:

$$\mathrm{y(n)} = k_i(1-\beta_i) * \mathrm{x(n)} + \beta_i * \mathrm{y(n-1)} \quad (4.26)$$

O filtro de correcção é um filtro FIR de primeira ordem com a seguinte função de transferência:

$$H_i(z) = g \frac{1-\beta z^{-1}}{1-\beta} \quad (4.27)$$

Cuja equação diferencial é:

$$y(n) = \frac{g(x(n) - \beta * x(n-1))}{1-\beta} \quad (4.28)$$

Os parâmetros para estes filtros são baseados no tempo de reverberação à frequência zero e frequência de *Nyquist*($T_{r0}(w), T_{r\pi}(w)$):

$$k_i = 10^{-3\tau_i/T_r(0)} \quad (4.29)$$

$$\beta_i = 1 - \frac{2}{1+k_i^{(1+1/\epsilon)}}, \qquad \beta = \frac{1-\sqrt{\epsilon}}{1+\sqrt{\epsilon}} \quad (4.30)$$

$$\epsilon = \frac{T_r(\pi)}{T_r(0)} \quad (4.31)$$

Existem várias propostas para definir o atraso de cada linha. A primeira proposta analisada consistiu na fórmula fornecida por Schroeder [Moorer, 1987] e que define o tempo em que as reflexões deixam de ser perceptíveis como ecos separados:

$$t_c = 5*10^{-5}\sqrt{V/\Delta t} \quad (4.32)$$

Em que V é o volume da sala e Δt impulso de excitação. No entanto como não está definido correctamente qual o valor a inserir como impulso de excitação decidiu-se analisar as outras possibilidades. A proposta seguinte consistia no método do *"mean free path"* que determina o tempo médio para uma onda sonora viajar de extremo a extremo da sala [Smith, 2008].

$$d = 4*T*c*\frac{V}{S} \quad (4.33)$$

em que V é o volume da sala, S a área de toda a superfície da sala, T o período de amostragem e c a velocidade do som no ar. Desta forma a resposta da FDN iria ser gerada tendo em conta o tempo estimado das primeiras reflexões. A última proposta analisada foi proposta em [Zölzer, 2002] e consiste em inserir como tempo de atraso o valor do atraso real das primeiras reflexões. Assim o resultado obtido é bastante mais real de que apenas usando valores estimados.
Como as primeiras reflexões vão ser geradas externamente ao algoritmo da reverberação tardia, ao usar estas duas últimas propostas, as primeiras amostras do resultado da FDN vão ser descartadas. Como o número de reflexões geradas pela FDN vai aumentando exponencialmente ao longo do tempo, ao usar o atraso das primeiras reflexões, garante-se que na transição entre as primeiras reflexões e a reverberação tardia, já esteja a ser gerado um número bastante elevado de reflexões.

4.3.4.1 Matriz de realimentação

A parte mais delicada na criação de uma estrutura FDN *lossless* consiste na escolha da matriz de realimentação [Zölzer, 2002]. Como foi referido anteriormente essa matriz tem que ser unitária. Uma das possíveis escolhas consiste em escolher uma Matriz do tipo *Householder* [Gardner, 1998] que verifica:

$$A_N = J_N - \frac{2}{N} U_N U_N^T \quad (4.34)$$

em que *Jn* é uma matriz de permutação N*X e U_N um vector contendo uns.
Jot descobriu que esta estrutura produz um eco parasita periódico com um período igual à soma dos atrasos de todas as linhas. Esse eco é o resultado entre a interferência construtiva entre as saídas das linhas de atraso e pode ser eliminado através do uso do parâmetro C_N ilustrado na figura 39 para que os outros canais sofram uma inversão de fase (multiplicação por -1). Para uma FDN com 6 linhas de atraso obtêm-se então a matriz:

$$C = [1, -1, 1, -1, 1, -1] \quad (4.35)$$

Existem outras matrizes que não foram analisadas, apesar de interessante, como as matrizes circulares a ter em conta em trabalho futuro, referidas na literatura com resultados bastante positivos [Gardner, 1998] [Zölzer, 2002] [Rochesso, 1997].

4.3.4.2 Resultados Obtidos

Para confirmar a eficácia das FDN geradas foram feitas algumas medições com o Aurora (ver Anexo 1) [Aurora, 2008] das respostas obtidas através de algumas das unidades de reverberação implementadas baseadas em FDN.
Os testes foram efectuados com uma matriz de realimentação *Householder*. Foram usadas 6 linhas de atraso em todos os testes. Nos primeiros testes foi usada uma FDN cujas perdas de absorção eram simuladas através de uma constante (fórmula de ganho proposta por Schroeder). O primeiro teste passou por inserir um tempo de reverberação igual em todas as linhas de atraso e, de seguida, gravar a resposta impulsional da FDN e verificar se o valor obtido com o Aurora coincidia com o valor inserido na unidade de reverberação. Os valores medidos encontram-se na tabela seguinte:

RT60 inserido na unidade[s]			5	5	5	5	5	5	5		
Freq. [Hz]	31.5	63	125	250	500	1000	2000	4000	8000	16000	Média
RT60 medido[s]	5.01	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00

Tabela 3 - Valores Obtidos no primeiro teste

Constata-se que o tempo de reverberação gerado pela FDN corresponde ao valor pretendido.

O teste seguinte consistiu em substituir o ganho que determinava a absorção do sinal pelos filtros de primeira ordem propostos por Jot. Os tempos de reverberação usados foram 3 seg. e 2 seg. para 0 Hz e 22050 Hz respectivamente. Os resultados obtidos através do Aurora foram os seguintes:

Freq.[Hz]	31.5	63	125	250	500	1000	2000	4000	8000	16000	A
RT60 [s]	3.00	3.00	3.00	3.00	3.00	2.99	2.95	2.83	2.52	2.12	2.47

Tabela 4 – Valores obtidos no segundo teste

Através do uso dos filtros é possível obter um decaimento maior nas frequências mais elevadas, como se comprova com os resultados obtidos, simulando assim a absorção do ar para as altas frequências. No entanto, apenas é possível definir tempos de reverberação para 0 e 22050 Hz, não sendo possível definir valores específicos para bandas de frequência distintas.

Tentou-se ainda inserir filtros passa banda em cada linha de atraso da FDN para assim ser zpossível simular o tempo de reverberação para cada banda separadamente. No entanto, os resultados não foram satisfatórios obtendo-se não um tempo específico para cada banda mas sim o tempo de reverberação mais elevado de todos os valores inseridos. Isso deve-se ao facto das linhas de atraso serem realimentadas entre si. Por exemplo, assumindo uma linha de atraso programada para fornecer um tempo de reverberação de 2s e 3 linhas programadas para um tempo de reverberação de 1s, o tempo de reverberação final vai ser igual a 2s devido à realimentação gerada pela linha de atraso programada com esse tempo de reverberação. Isto é verdade se os filtros passa-banda forem inseridos após o ciclo de realimentação (conforme ilustra a figura 39). Se os filtros forem inseridos antes do ciclo de realimentação, cada linha de atraso apenas permite passar frequências numa banda distinta. Dessa forma, o ciclo de realimentação não faria sentido pois os valores realimentados entre linhas de atraso distintas iriam ser sempre atenuados devido aos filtros passa banda, sendo apenas aproveitado-se em cada linha de atraso os valores gerados por essa linha. Neste caso, estaria-se perante um algoritmo Multirate (ver ponto seguinte).

Os algoritmos baseados em FDN têm uma resposta bastante natural. Se o objectivo apenas consiste na simulação de um tempo de reverberação médio, ignorando o tempo especifico por bandas então este é um algoritmo bastante eficiente e com um custo computacional relativamente baixo. No entanto, tem a limitação de não permitir a simulação de tempos de reverberação distintos por banda. Decidiu-se estudar mais uma unidade de reverberação capaz de ultrapassar essa limitação.

4.3.5 Algoritmo Multirate

Uma alternativa ao algoritmo de Jot consiste em combinar n filtros passa-banda com filtros comb de tal forma que cada filtro comb processe uma gama de frequências distintas. O ganho do filtro comb determina o tempo de reverberação para a banda de frequência correspondente. Desta forma é possível obter um algoritmo de reverberação que simule o tempo de reverberação específico por bandas.

4.3.5.1 Implementação dos filtros

Os primeiros filtros implementados consistiram numa classe especial de filtros IIR (*infinite impulse response*) parametrizáveis (all-pass, passa-baixo, passa-banda, passa-alto e rejeita-banda). Estes filtros são relativamente fáceis de sintonizar, sendo para isso apenas necessário alterar um ou dois coeficientes. Desempenham ainda um papel importante em aplicações em tempo real dado a sua baixa complexidade computacional. Como estamos perante filtros IIR a saída do filtro irá ser realimentada novamente para a entrada. Isso acontece devido aos ciclos de realimentação da estrutura, em que um impulso de entrada gera um sinal de saída cuja duração é dependente dos coeficientes do filtro, podendo-se estender durante longos períodos de tempo. Foram estudadas também outras estruturas sem os ciclos de realimentação (denominados por filtros FIR – *Finite Impulse Response*) [Zölzer, 2002]. Nos pontos seguintes está descrito cada conjunto de ciclos analisados.

4.3.5.1.1 Filtros IIR parametrizáveis

O filtro base para a implementação desta gama de filtros IIR consiste no filtro *all-pass*. O filtro all-pass descrito no ponto 414.3.1 é um filtro de primeira ordem em que o parâmetro g define o tempo de decaimento do sinal de entrada. No entanto, é possível ajustar os parâmetros do filtro em relação a uma frequência de corte específica e, caso o filtro a implementar seja de segunda ordem, a sua largura de banda. De seguida encontra-se uma descrição mais detalhada dos filtros analisados.

Filtros parametrizáveis de primeira ordem

A função de transferência de um filtro *all-pass* de primeira ordem é:

$$A(z) = \frac{z^{-1} + g}{1 + gz^{-1}} \quad (4.36)$$

obtendo assim a seguinte equação diferencial:

$$y(n) = gx(n) + x(n-1) - gy(n-1) \quad (4.37)$$

em que g define o ganho do filtro e, através do mesmo, o tempo de decaimento do filtro. No entanto, através da fórmula seguinte é possível determinar o ganho tendo em conta uma frequência de corte específica:

$$g = \frac{\tan(\pi f_c / f_s) - 1}{\tan(\pi f_c / f_s) + 1} \quad (4.38)$$

em que f_c é a frequência de corte e f_s *a* frequência de amostragem.

Com o uso deste filtro é possível implementar filtros passa-baixo e passa-alto de primeira ordem. A função de transferência do filtro passa-baixo/alto é então dada por:

$$H(z) = \frac{1}{2}\left(1 \pm A(z)\right) \quad (4.39)$$

O sinal de adição (+) indica que se está perante um filtro passa-baixo e o sinal de subtracção (-) perante um filtro passa-alto.

Para criar um filtro passa banda de primeira ordem adiciona-se um filtro passa baixo em paralelo com um filtro passa-alto, criando então assim o filtro passa banda pretendido.

Na tabela em baixo é possível ver o espectograma dos vários filtros de primeira ordem criados. O sinal de entrada dos filtros foi ruído branco.

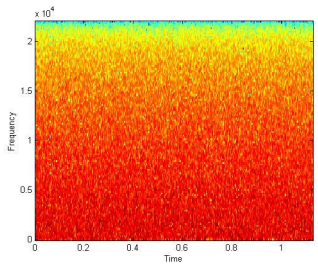
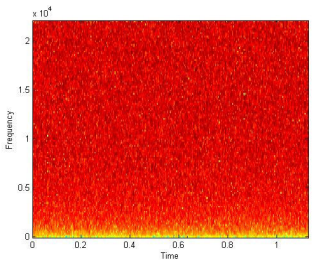
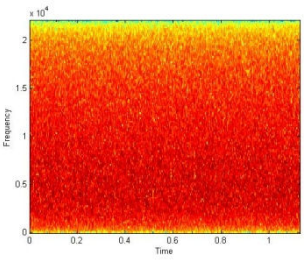
Filtro passa baixo (frequência de corte 5000 Hz)	Filtro passa alto (frequência de corte 5000 Hz)	Filtro passa banda (banda [4000, 5000] Hz)
		

Figura 40 - Resultado dos filtros IIR de primeira ordem criados

Como é possível verificar pelas figuras anteriores a atenuação dos filtros não é muito eficiente, sendo a banda de transição muito larga.

Filtros parametrizáveis de segunda ordem

A função de transferência de um filtro *all-pass* de segunda ordem é:

$$A(z) = \frac{-c + d(1-c)z^{-1} + z^{-2}}{1 + d(1-c)z^{-1} - cz^{-2}} \quad (4.40)$$

obtendo assim a seguinte equação diferencial:

$$y(n) = -cx(n) + d(1-c)x(n-1) + x(n-2)$$
$$-d(1-c)y(n-1) + cy(n-2) \quad (4.41)$$

As constantes *d* e *c* definem a frequência de corte e a largura de banda respectivamente. Esses parâmetros são calculados pelas seguintes fórmulas:

$$d = \cos(-2\pi f_c/f_s) \quad (4.42)$$

$$c = \frac{\tan(\pi f_b/f_s) - 1}{\tan(\pi f_b/f_s) + 1} \quad (4.43)$$

em que f_c é a frequência de corte, f_b a largura de banda e f_s *a* frequência de amostragem.

Através do filtro *all-pass* de segunda ordem é possível criar o filtro passa-banda e rejeita-banda. A função de transferência para a implementação dos mesmos é dada por:

$$H(z) = \frac{1}{2}\left(1 \pm A(z)\right) \quad (4.44)$$

O sinal de adição (+) indica que se está perante um filtro rejeita-banda e o sinal de subtracção (-) que estamos perante um filtro passa-banda. A(z) é a função de transferência do filtro *all-pass.*

Os filtros parametrizáveis de segunda ordem passa baixo e passa alto são obtidos através da seguinte função de transferência:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} - a_2 z^{-2}} \quad (4.45)$$

Cuja equação diferencial é:

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$
$$-a_1 y(n-1) + a_2 y(n-2) \quad (4.46)$$

Os coeficientes dos filtros são obtidos através das fórmulas descritas no quadro seguinte:

Filtro Passa baixo de segunda ordem ($K = \tan(\pi f_b/f_s)$)				
$b_0 = \frac{K^2}{1+\sqrt{2}K+K^2}$	$b_1 = \frac{2K^2}{1+\sqrt{2}K+K^2}$	$b_2 = \frac{K^2}{1+\sqrt{2}K+K^2}$	$a_1 = \frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$a_2 = \frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Filtro Passa alto de segunda ordem ($K = \tan(\pi f_b/f_s)$)				
$b_0 = \frac{1}{1+\sqrt{2}K+K^2}$	$b_1 = \frac{-2}{1+\sqrt{2}K+K^2}$	$b_2 = \frac{1}{1+\sqrt{2}K+K^2}$	$a_1 = \frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$a_2 = \frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$

Tabela 5 - Fórmulas para determinar os coeficientes de segunda ordem de filtros IIR

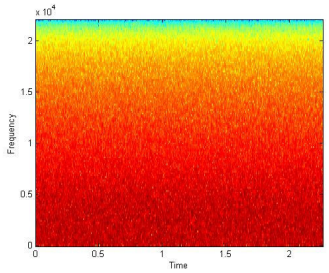
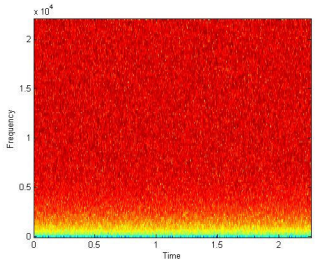
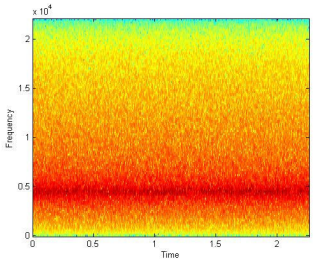
Filtro passa baixo (frequência de corte 5000 Hz)	Filtro passa alto (frequência de corte 5000 Hz)	Filtro passa banda (banda [4000, 5000] Hz)
		

Figura 41 - Resultado dos filtros IIR de segunda ordem implementados

A resposta melhora consideravelmente, comparando com os filtros parametrizáveis de primeira ordem. No entanto, mesmo assim a resposta ainda não é muito eficiente, especialmente nos filtros passa alto e passa baixo.

Conexão de vários filtros parametrizáveis em série

Uma das formas possíveis para melhorar a eficácia dos filtros é unir vários filtros em série. Como teste, inseriu-se 3 filtros em série, um de 1ª ordem e os dois seguintes de 2ª ordem.

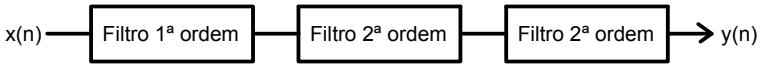


Figura 42 - Filtros parametrizáveis ligados em série

Os resultados obtidos encontram-se descritos na figura seguinte:

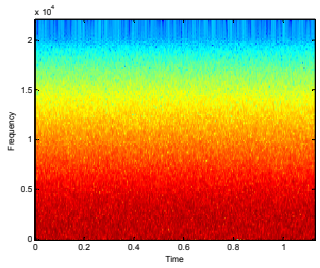
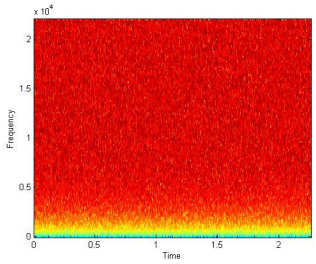
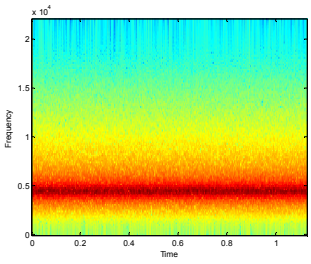
Filtro passa baixo (frequência de corte 5000 Hz)	Filtro passa alto (frequência de corte 5000 Hz)	Filtro passa banda (banda [4000, 5000] Hz)
		

Figura 43 - Resultado dos Filtros IIR ligados em série

Com este método já se obtêm resultados satisfatórios. É possível verificar que os cortes efectuados são extremamente precisos e podem ser melhorados se o número de filtros em paralelo inseridos for maior.

4.3.5.1.2 Filtros FIR

Os filtros descritos no ponto anterior (passa banda, passa-baixo, passa-alto e rejeita-banda) também podem ser efectuados através de estruturas FIR. Para isso é necessário calcular a resposta impulsional do filtro. A estrutura do filtro mais simples é baseada na transformada inversa de Fourier em tempo discreto que resulta no filtro passa baixo ideal. A sua resposta impulsional é:

$$h_{LP}(n) = \frac{2f_c}{f_s}\frac{\sin\left[2\pi f_c/f_s\left(n-\frac{N-1}{2}\right)\right]}{\pi f_c/f_s\left(n-\frac{N-1}{2}\right)}, n = 0, \dots, N-1 \quad (4.47)$$

De forma a melhorar a resposta em frequência este impulso pode ser pesado por uma janela como *Hamming* ou *Blackman* da seguinte forma:

$$h_B(n) = h(n).w_B(n) \quad (4.48)$$

$$h_H(n) = h(n).w_H(n) \quad (4.49)$$

$$, n = 0, \dots, N-1$$

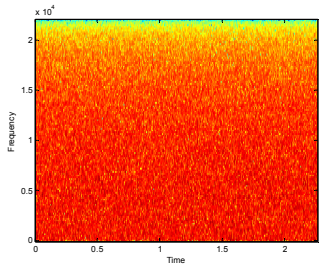
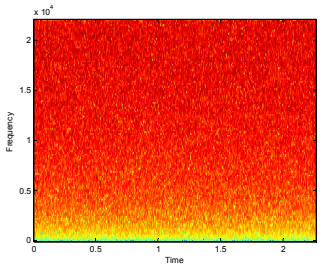
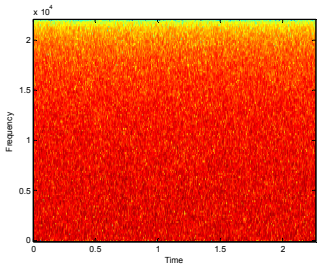
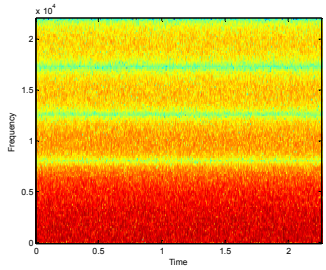
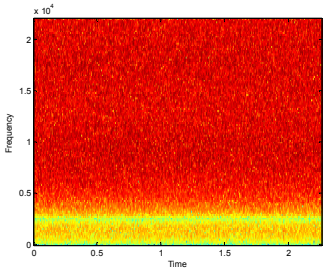
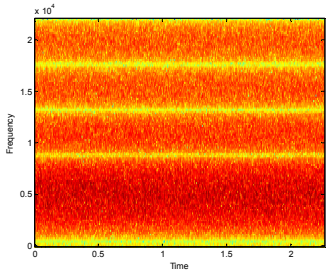
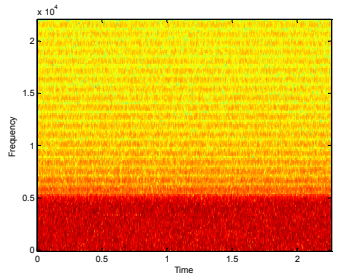
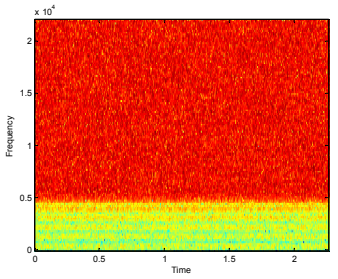
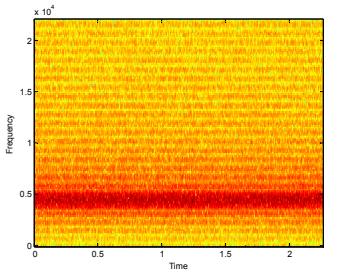
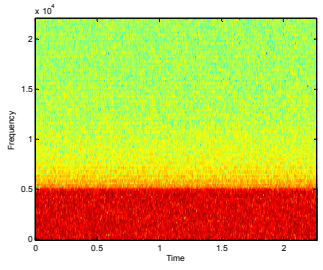
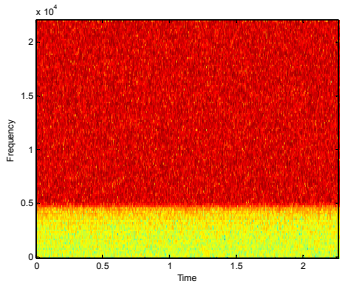
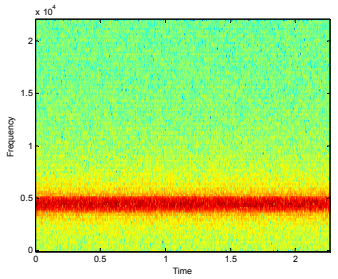
A partir da transformação da resposta impulsional de um filtro passa-baixo é possível obter um filtro passa alto ou passa banda. Essas transformações, no domínio do tempo, são efectuadas pelas seguintes equações.

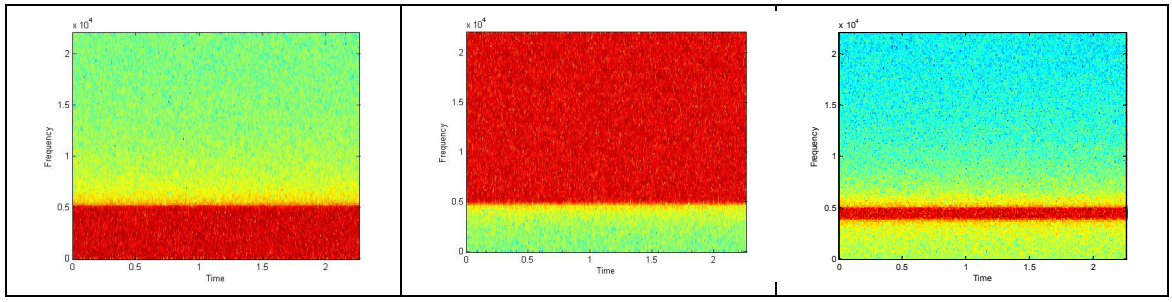
- Passa alto

$$h_{HP}(n) = h_{LP}(n).\cos\left[\pi\left(n-\frac{N-1}{2}\right)\right], \mathrm{n} = 0, \dots, \mathrm{N}-1 \quad (4.50)$$

- Passa banda

$$h_{BP}(n) = 2h_{LP}(n).\cos\left[2\pi f_c/f_s\left(n - \frac{N-1}{2}\right)\right], \mathrm{n} = 0, \ldots, \mathrm{N}-1 \qquad (4.51)$$

Filtro passa baixo (freq. de corte 5000 Hz)	Filtro passa alto (freq. de corte 5000 Hz)	Filtro passa banda (banda [4000, 5000] Hz)
Filtro FIR com 2 amostras		
		
Filtro FIR com 10 amostras		
		
Filtro FIR com 50 amostras		
		
Filtro FIR com 100 amostras		
		
Filtro FIR com 200 amostras		



Através dos espectrogramas anteriores é possível determinar que através do uso de filtros FIR com 100 amostras na sua resposta impulsional obtêm-se bandas de corte bastante precisos, não existindo quase valores de outras frequências que não as permitidas pelos filtros.

4.3.5.2 Resultados e conclusões

Todos os filtros descritos no ponto anterior foram analisados de forma a determinar quais os melhores em termos de qualidade/custo computacional. Os filtros FIR são extremamente eficazes. No entanto, como o seu uso implica calcular a convolução de um som com a sua resposta impulsional têm um peso computacional mais elevado.

Foram usados os filtros FIR descritos no ponto anterior para implementar o algoritmo Multirate. De seguida, foram efectuados testes para verificar se os tempos de reverberação por bandas estavam correctos. Os primeiros testes consistiam em activar apenas uma banda de frequência específica, ou seja, fornecer um tempo de reverberação não nulo numa banda e nulo nas restantes. Para verificar se o algoritmo fornece o tempo de reverberação correcto é necessário obter a resposta impulsional do filtro e medir o seu tempo de reverberação. O tempo de reverberação foi medido pelo Aurora.

Ao inserir no algoritmo os seguintes tempos de reverberação: [6,0,0,0,0,0] segundos, correspondendo às bandas de frequência [125,250,500,1000,2000,4000] Hz, os resultados obtidos através do Aurora foram os seguintes:

Freq. [Hz]	31.5	63	125	250	500	1000	2000	4000	8000	16000
T30 [s]	6.07	6.02	6.50	6.31	6.57	11.40	0.01	0.00	0.00	0.00

Tabela 6

Como é possível verificar, obteve-se um valor perto de 6 segundos na banda dos 125 Hz. No entanto obteve-se também valores inesperados como os 11.40 segundos na banda dos 2000 Hz, quando o valor de entrada era nulo (zero segundos).

Ao inserir os seguintes tempos de reverberação: [0,0,0,0,0,6] os resultados obtidos pelo Aurora são:

Freq. [Hz]	31.5	63	125	250	500	1000	2000	4000	8000	16000
T30 [s]	5.97	6.10	6.12	6.14	6.14	6.12	5.94	5.93	5.94	5.94

Tabela 7

Confirma-se que o tempo de reverberação para 4000 Hz é igual ao inserido. No entanto, os tempos de reverberação para as frequências mais baixas não são nulos como previsto. O espectograma da resposta obtida é:

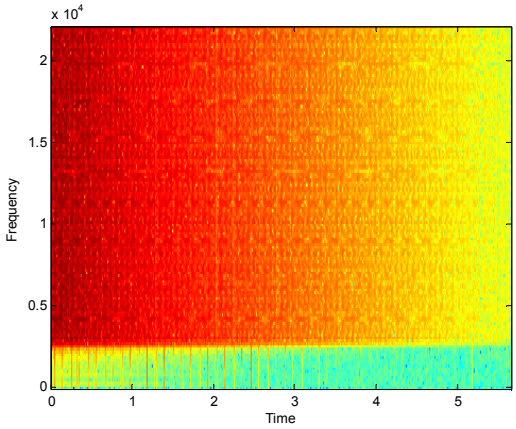


Figura 44 - Exemplo do espectrograma de saída do algoritmo de reverberação

Através da análise do espectograma verifica-se que não estão presentes as frequências entre 0 e 2500 Hz aproximadamente, ou seja nota-se uma discrepância entre os valores medidos pela aurora e os valores ilustrados pelo espectograma. Como não existe informação disponível sobre os cálculos que o Aurora efectua não é possível determinar qual o motivo dessa distinção de valores.

O passo seguinte consistiu em inserir vários tempos de reverberação distintos por bandas e verificar o resultado fornecido pelo Aurora. Os tempos de reverberação inseridos foram [7,0,0,3,0,5]. Os resultados obtidos foram:

Freq. [Hz]	31.5	63	125	250	500	1000	2000	4000	8000	16000
T30 [s]	6.98	6.98	7.38	7.24	4.44	3.02	4.94	4.97	4.97	4.97

Tabela 8

O espectograma do resultado é:

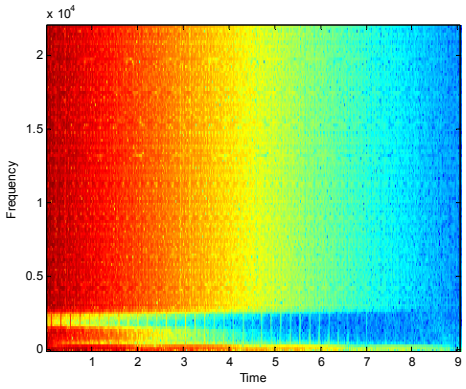


Figura 45 -Resultados obtidos pelo Aurora

Os tempos de reverberação medidos pelo Aurora estão de acordo com os tempos de reverberação não nulos inseridos no algoritmo. No entanto, como no exemplo anterior o Aurora deu tempos de reverberação positivos para bandas de frequência que não estão activas (como é possível verificar pelo espectograma anterior).

último teste efectuado consistiu em aumentar o número de bandas não nulas. Os tempos de reverberação usados foram [7,1,6,2,5,2]. Os resultados obtidos foram:

Freq. [Hz]	31.5	63	125	250	500	1000	2000	4000	8000	16000
T30 [s]	6.98	6.98	7.37	7.18	3.88	3.02	4.94	4.97	4.97	4.96

Tabela 9

Como é possível verificar, os tempos de reverberação medidos pelo Aurora já não estão de acordo com os valores inseridos no algoritmo estando apenas a banda dos 125 Hz com um valor aproximado do inserido. No entanto através da análise do espectograma verifica-se que os cortes das frequências estão a ser efectuados correctamente (figura 46). De notar especialmente a banda entre os 4000 Hz e 22050 Hz cujo tempo no espectograma não ultrapassa os 3 segundos

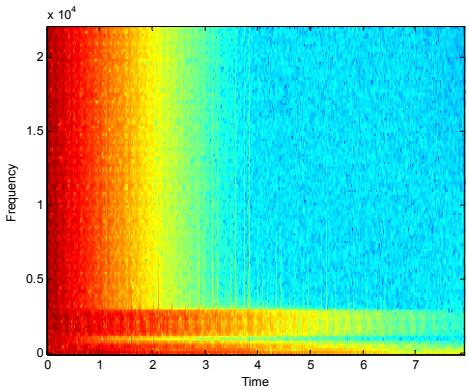


Figura 46 - Resultados obtidos pelo Aurora

As frequências de corte efectuadas no algoritmo *Multirate* são determinadas da mesma forma às usadas quando os coeficientes de absorção dos materiais são aplicados (Figura 29). Provavelmente as frequências de corte usadas no Aurora são diferente e daí resultam valores diferentes. No entanto este é um ponto em aberto que necessita de um estudo mais aprofundado.

5 Resultados

5.1 3DLIB

Foi criada a biblioteca 3DLIB para gerar som 3D tendo em conta vários parâmetros que podem ser definidos pelo utilizador. Esta biblioteca resulta da união de todos os algoritmos analisados nos capítulos anteriores e foi criada para que a auralização em ambientes virtuais seja efectuada de uma forma simples e rápida. De seguida encontram-se descritos os comandos necessários para usufruir das suas funcionalidades.

O primeiro passo para a usar consiste em definir a posição do utilizador durante a experiência. Isso é efectuado através dos seguintes comandos:

```
user *util = new user;
util->SetPosition(posUtil);
```

Em que *user* é a classe que vai conter informação sobre o utilizador e *posUtil* um vector com 3 elementos que contêm a posição do utilizador. O próximo passo passa por definir a sala a ser usada. A variável *nomeSala* define o nome do objecto da sala.

```
room *sala = new::room(nomeSala);
```

De seguida é necessário criar a fonte sonora da seguinte forma:

```
sound * som = new sound;
som->OpenSound("c:\\sonsTeste\\drum44100.wav");
```

Para definir a posição no espaço de um som é necessário definir um canal com esse som. O canal é criado com o código seguinte:

```
channel * canal = new channel;
canal->SetChannel(som,posFonte,LOOP);
```

A variável *posFonte* contêm a posição da fonte na sala. O terceiro parâmetro define se o som vai ser efectuado em *loop* ou não (foram definidas constantes LOOP e NO_LOOP para facilitar o seu uso). O próximo passo consiste em inicializar os dados. Isso é efectuado da seguinte forma:

```
strMain = new stream();
strMain->initialize(freqAm,bufSize,util,sala,ordem, revAlg,
float * RT60);
```

em que *freqAm* define a frequência de amostragem, *bufSize* o número de amostras que vai ser analisado em cada invocação da *callback* do PA. *util* o objecto com informação sobre o utilizador, *sala* o caminho até ao objecto 3D da sala a simular, *ordem* a ordem das reflexões a calcular(1ª ou 2ª ordem), *revAlg* o algoritmo de reverberação a usar e *RT60* o tempo de reverberação a usar no algoritmo de reverberação (ou um vector com os tempos, dependendo do algoritmo pretendido).

Num Pentium Duo T2300 com 1 GB de RAM, conseguiu-se até reproduzir 36 fontes distintas (som directo e reflexões) sem que o som de saída fosse prejudicado.

O algoritmo de reverberação é escolhido tendo em conta as seguintes constantes (P_COMB- filtro comb em paralelo, FDN - *Feedback Delay Networks*, MULTIRATE – algoritmo *Multirate*).

Para activar o canal efectua-se o seguinte comando:

```
strMain->startChannel(canal);
```

Agora é necessário usar o comando seguinte dentro da *callback* do PA para obter o som calculado já com reflexões, reverberação e espacialização:

```
strMain->SetOutput(out);
```

Por fim, caso seja permitido ao utilizador interagir com o ambiente é necessário actualizar os seus parâmetros. O comando usado para actualizar o utilizador é:

```
util->updateListener(pitch,yaw,roll);
```

em que *pitch, yaw* e *roll* são os novos ângulos da cabeça do utilizador.

5.2 Testes perceptuais

Para verificar a capacidade de o utilizador determinar a posição espacial das fontes sonoras usando a biblioteca criada foram efectuados os seguintes testes perceptuais:

- Localização no plano horizontal/plano vertical
    - Apenas som directo
        - Utilizador estático
        - Utilizador dinâmico
    - Com primeiras reflexões
        - Utilizador estático
        - Utilizador dinâmico

Na figura 47 encontra-se uma imagem do teste efectuado no plano horizontal com as primeiras reflexões activas e com o utilizador estático. As primeiras reflexões foram calculadas tendo em conta a sala apresentada. Os objectos que se encontram na figura eram coloridos para ser mais fácil o utilizador indicar qual a posição que considera correcta para a fonte sonora. Nos testes em que o utilizador era estático perante o mundo não foi fornecida informação visual.

Figura 47 - Teste perceptual efectuado no plano horizontal (primeiras reflexões activas)

Na tabela seguinte são fornecidos alguns exemplos dos resultados obtidos durante os testes perceptuais no plano horizontal (apenas som directo). O ponto em cada imagem indica a posição correcta das fontes e os raios indicam o ângulo escolhido pelos utilizadores. Verifica-se que a capacidade de localização melhora nos testes em que o utilizador pode interagir com o mundo. No entanto, é necessário ter em conta o problema da "confusão frente-trás" quando estamos perante o utilizador estático, ou seja, se o utilizador não puder interagir com o mundo pode ter dificuldades em verificar se a fonte sonora se encontra à sua frente ou atrás de si [Kendall,

1995]. Em relação aos testes efectuados no plano vertical nestas condições, a capacidade para o utilizador determinar a origem do som é bastante reduzida.

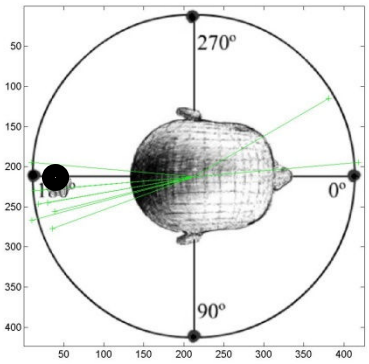
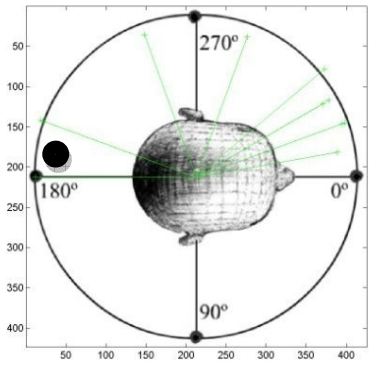
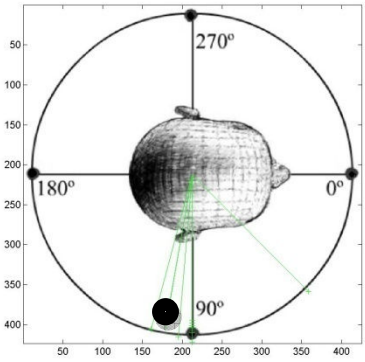
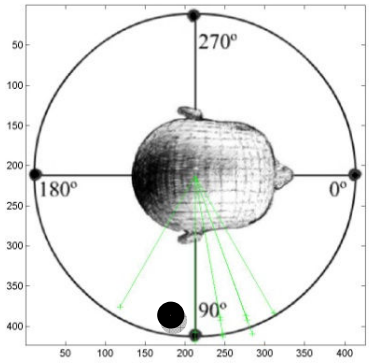
Utilizador dinâmico	Utilizador estático
	
	

Figura 48 - Resultados dos testes perceptuais

No entanto, as experiências foram efectuadas com um grupo de 11 utilizadores, pelo que os resultados não são muito conclusivos. Apesar do número reduzido de utilizadores, parece confirmar-se a má capacidade de localização do ser humano para fontes no plano horizontal e a melhoria na localização da posição das fontes sonoras quando o utilizador interage com o mundo. Como as experiências já estão criadas, é possível agora efectuar os testes adicionais mais utilizadores para obter valores mais precisos.

5.3 Demonstração

Para verificar a eficácia da biblioteca criada foi gerado um programa de demonstração que simula um concerto virtual. Funciona como uma curta-metragem: foram colocadas numa sala vários sons e personagens (modelos 3D) distintos que vão sendo activados ou desactivados de acordo com o desenrolar do espectáculo. A posição da fonte é alterada tendo em conta os valores lidos pelo sensor *InterTrax*. Nesta demonstração, o utilizador mantém-se numa posição fixa, tal como aconteceria numa sala de espectáculo real, onde apenas é possível mover livremente a cabeça. Apenas foram calculadas reflexões de primeira ordem e a reverberação tardia foi calculada através das FDN com um tempo de reverberação igual em todas as linhas de atraso e igual a 1,5 segundos.

Inicialmente entra em palco a anfitriã (Figura 50) para apresentar a banda que vai "subir" ao palco. De seguida, a actuação dos músicos começa (Figura 49). Quando a actuação termina, a anfitriã vem agradecer ao espectador a sua presença, e por fim, entra em cena o nosso *"big ear sponsor"* recitando um pequeno monólogo do *Timothy Leary* (Figura 51). À excepção do som dos músicos, os sons foram criados recorrendo a um sintetizador de voz. Os sons vão sendo activados e desactivados tendo em conta o desenrolar do espectáculo. A interacção com os óculos e o sensor InterTrax foram geradas da mesma forma que no ponto 4.2.2.

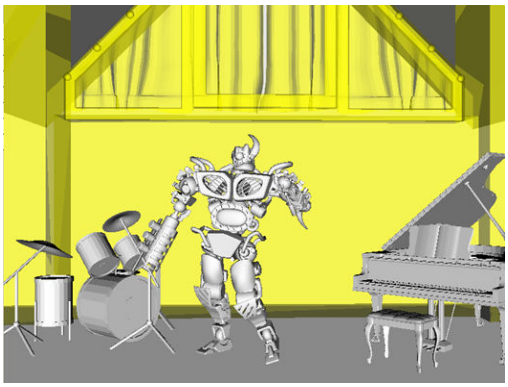


Figura 49 - Músicos do espectáculo

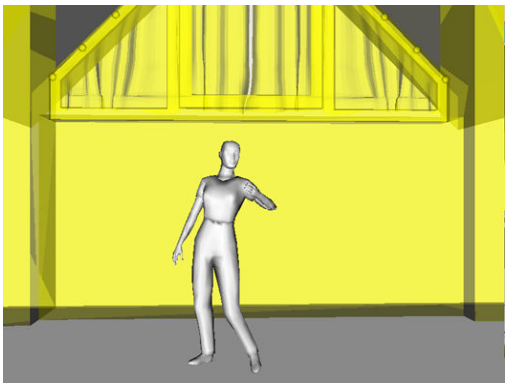


Figura 50 - Anfitriã do espectáculo

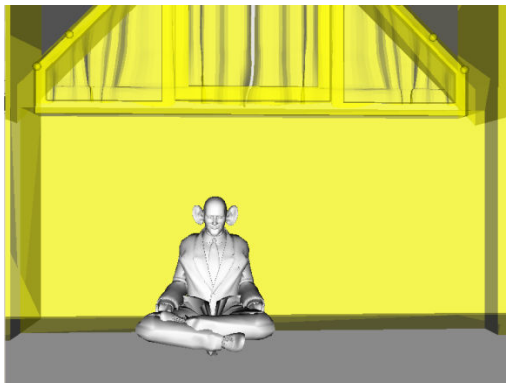


Figura 51 - Big-ear Sponsor

6 Conclusões e trabalho futuro

Com este trabalho, foi criada a biblioteca 3DLIB que gera som 3D de uma forma eficaz em tempo real abstraindo o utilizador do processamento efectuado.

Foi implementado um algoritmo para determinar as primeiras reflexões, através do método das imagens virtuais, tendo como entrada o modelo 3D de uma sala. Os resultados obtidos por esse algoritmo contêm informação sobre a posição das fontes virtuais de uma fonte real e o *ID* (valor numérico que é distinto para cada célula) das células do modelo que definiram cada fonte virtual. Conhecendo o coeficiente de absorção dos materiais de cada célula (através do programa de correcção acústica) é possível calcular o som da fonte virtual atenuando o som da fonte real de acordo com os coeficientes de absorção das células onde ocorreu a reflexão. Obtém-se assim uma resposta para as primeiras reflexões muito próxima da realidade, apesar de o algoritmo ter um peso computacional elevado. No entanto, como os cálculos podem ser efectuados *off-line*, o tempo de processamento não é crítico. Além disso, é possível guardar os resultados, caso estes tenham que ser usados em diferentes ocasiões.

Foram implementados vários algoritmos para simular a reverberação tardia. Através da criação da resposta impulsional sintética é possível obter um som reverberante bastante natural, sendo um dos algoritmos a ter em consideração caso o cálculo não necessite de ser efectuado em tempo real.

Os algoritmos para criar reverberação em tempo real a realçar são as FDN e o algoritmo Multirate. Através da FDN é possível simular um tempo de reverberação médio bastante preciso em que o resultado é bastante natural, com a vantagem do seu peso computacional ser relativamente pequeno.

Como a implementação do algoritmo Multirate se baseia em filtros passa-banda, foi efectuada uma análise de filtros IIR e FIR para verificar qual o melhor em termos de peso computacional e banda de transição. Dos filtros analisados os mais eficazes são os filtros FIR. Contudo, a melhor escolha em termos de qualidade/peso computacional, em particular para aplicações em tempo real, são os filtros IIR ligados em série.

Quando se usa o algoritmo Multirate, os valores medidos experimentalmente com o Aurora não coincidem com os valores inseridos na unidade de reverberação. No entanto, como este é o único algoritmo que consegue fornecer um tempo de reverberação distinto por banda em tempo real, deve ser estudado de forma mais aprofundada no futuro. Todavia, o algoritmo Multirate tem as desvantagens de ser mais pesado computacionalmente que as FDN, se forem usados filtros FIR de ordem elevada para fazer a divisão por bandas, e de não conseguir gerar uma quantidade de reflexões tão elevada.

Estão implementados vários testes perceptuais que podem ser usados no futuro para determinar a eficácia dos algoritmos implementados. Embora não tenham sido efectuados testes

envolvendo a reverberação tardia existem vários sons já criados que podem ser usados para esse fim.

Foi gerado um programa de demonstração para verificar a funcionalidade da biblioteca 3DLIB, com resultados satisfatórios. Esse programa de demonstração já foi apresentado em várias ocasiões, tendo ganho o primeiro prémio do *"Concurso de Realizações Tecnológicas em Engenharia de Áudio"* promovido no 9º Encontro Anual da Associação Portuguesa de Engenharia de Áudio (APEA) realizado o Instituto Politécnico de Leiria em 20 de Outubro de 2007.

Como trabalho futuro é possível analisar outras matrizes de realimentação da FDN, como as matrizes circulares, e novos algoritmos de reverberação como os '*digital waveguide mesh'*. Os testes perceptuais devem ser efectuados com um leque maior de utilizadores. O algoritmo Multirate necessita de ser investigado de forma mais aprofundada para compreender a discrepância entre os tempos de reverberação inseridos no algoritmo e os tempos medidos através do Aurora. As funcionalidades implementadas na biblioteca 3DLIB apenas correspondem a funções sobre acústica arquitectural. No entanto, como já estão implementados vários filtros IIR, FIR e vários algoritmos de reverberação é possível ampliar o uso da biblioteca para gerar efeitos sonoros, estendendo assim as possibilidades de uso às mais variadas áreas e não só a ambientes de realidade virtual.

Anexo 1 Material de apoio

1 Software

Matlab

O MATLAB [Matlab, 2008] é uma ferramenta direccionada para o cálculo numérico. Algumas das suas funcionalidades são análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos. Esta ferramenta permite resolver muitos problemas numéricos de uma forma mais rápida que a necessária para escrever um programa semelhante numa outra linguagem de programação, sendo ideal para efectuar protótipos. Esta ferramenta foi usada sempre que era necessário ter uma melhor noção de algum problemas específico e para determinar qual a melhor resolução para os mesmos de uma forma rápida.

Microsoft Visual C++

O Microsoft Visual C++ [VS2005, 2008] é uma ferramenta de desenvolvimento para desenvolver aplicações em C, C++ e C++/CLI desenvolvido pela Microsoft. É bastante fácil de usar, e oferece funcionalidades ao programador que permitem optimizar o seu trabalho, diminuindo assim o tempo de implementação dos projectos. Esta ferramenta vem incluída no Microsoft Visual Studio 2005.

Aurora

O Aurora consiste num conjunto de ferramentas para usar em conjunto com o *Cool Edit Pro* ou o *adobe Audition* que transforma um computador num aparelho de medição e reprodução comparável a um editor digital de ondas ou de um sistema de som binaural. Algumas das suas funcionalidades são medições do tempo de reverberação de resposta impulsionais de salas e a criação de som 3D.

Cool Edit Pro

O Cool Edit Pro é um programa para gravação e processamento de áudio digital com suporte *multitracking*. Permite a adição de novas ferramentas às já existentes no programa (como o conjunto de ferramentas fornecido pelo Aurora).

FMOD Ex

O sistema de Som FMOD Ex 0 é um motor de áudio não só para programadores de jogos mas também para designers de som, músicos e engenheiros de som. É um sistema extremamente intuitivo e fácil de usar fornecendo uma implementação de um ambiente áudio complexo para jogos e aplicações usando o mínimo de recursos possíveis.

Algumas das suas principais características são:

- Conjunto de DSP predefinido de efeitos especiais que não necessita de nenhuma plataforma nem de nenhum sistema operativo em específico.
- Suporte as novas gerações de consolas (PS3, Xbox360, Nintendo Wii, etc.)
- Novas funcionalidades e ferramentas que permitem que o FMOD possa ser usado por Designers de Som e músicos.
- Suporte completo para ambientes sonoros 3D incluindo vários métodos de atenuação, capacidade de múltiplos receptores sonoros, oclusão e obstrução (usando polígonos reais), Cones Sonoros.
- Fontes Sonoras virtuais que permitem a um jogo reproduzir milhares de sons de uma vez só sem nos termos de preocupar no problema de activar e desactivar a reprodução do som por si só.
- Suporta mais de 20 formatos de áudio diferentes.
- Motor de Streaming avançado
- Uma nova API orientada aos objectos suportado em C, C++, Delphi e Visual Basic

Libsndfile 1.0.17

Libsndfile 0 é uma biblioteca escrita em C para a leitura e escrita de ficheiros de som.

As acções principais desta biblioteca são:

- Capacidade de ler e escrever ficheiros em vários formatos diferentes.
- Uma interface simples, elegante e fácil de usar.
- Compatibilidade com Unix, Win32, MacOS entre outros.
- Conversão directa entre diferentes formatos,

- Normalização opcional ao ler dados em vírgula flutuante de ficheiros com os dados em *integer*.
- Capacidade de abrir ficheiros em modo leitura/escrita.
- Capacidade de abrir o cabeçalho do ficheiro sem fechar o mesmo (apenas em ficheiros abertos em modo leitura-escrita)

ASIO SDK

A biblioteca ASIO [ASIO, 2008](*Audio Stream I/O*) é usada para criar *Streams* de *Input/Output* de baixa latência. A interface não tem um número máximo definido de canais de entrada e saída estando esses valoras apenas limitados ao poder de processamento e ao *troughput* do sistema. Não define qualquer limitação da frequência de amostragem, formato das amostras (16, 24, 32 bit ou 32/*64 bit floating point*). Suporta soluções sofisticadas de hardware para processamento adicional mas mantém-se simples comparando-o a outras abordagens.

PortAudio

O *PortAudio*[PortAudio, 2008] (PA) é uma biblioteca escrita em C que fornece uma forma fácil de criar *streams* áudio. O seu objectivo principal é ser uma biblioteca *open source* que corra em várias plataformas diferentes para que assim os programas que a usam poderem correr em diferentes sistemas operativos. A forma como o PA cria a portabilidade entre vários sistemas operativos consiste em usar API de som específicas para cada Sistema, criando uma interface igual para sem usada em qualquer sistema operativo. Assim a única coisa necessária para executar uma aplicação em PA em vários sistemas operativos é compilar o PA com a API compatível do sistema operativo.

Compilação do PortAudio

Para podermos compilar o PA usando o ambiente de desenvolvimento Windows ASIO necessitamos de efectuar o *download* do ASIO SDK da *Steinberg*[ASIO, 2008].

Os passos para compilar o PA estão descritos nos pontos seguintes:

Fazer o #*include* do cabeçalho "*PortAudio.h*" no Projecto.

Adicionar os seguintes ficheiros *.cpp* relacionados com o *ASIO* ao projecto:

```
pa_asio.cpp                 (Portaudio\src\hostapi\asio)
asio.cpp                    (asio2sdk\common)
asiodrivers.cpp             (asio2sdk\host)
asiolist.cpp                (asio2sdk\host\pc)
```

Adicionar os seguintes ficheiros *.cpp* relacionados com o PortAudio ao projecto:

```
pa_allocation.c                (Portaudio\src\common)
pa_converters.c                (Portaudio\src\common)
pa_cpuload.c                   (Portaudio\src\common)
pa_dither.c                    (Portaudio\src\common)
pa_front.c                     (Portaudio\src\common)
pa_process.c                   (Portaudio\src\common)
pa_skeleton.c                  (Portaudio\src\common)
pa_stream.c                    (Portaudio\src\common)
pa_trace.c                     (Portaudio\src\common)
pa_win_hostapis.c              (Portaudio\src\os\win)
pa_win_util.c                  (Portaudio\src\os\win)
pa_x86_plain_converters.c      (Portaudio\src\os\win)
pa_win_wmme.c                  (Portaudio\src\hostapi\wmme)
```

O passo seguinte consiste em adicionar os seguintes *paths* ao projecto

```
portaudio\include
portaudio\src\common
asiosdk2\common
asiosdk2\host
asiosdk2\host\pc
```

Nas propriedades do projecto alterar o *character Set* de "*Use Unicode Character Set*" para "*Use Multi-Byte Character Set*"

O último passo é abrir o ficheiro "*pa_win_hostapis.c*" e adicionar as seguintes linhas:

```
#define PA_NO_MME
#define PA_NO_DS
```

Isto previne a inicialização do Windows MME e DirectSound(outras API compatíveis).

Para verificar se o PortAudio foi bem configurado convém compilar o ficheiro "*Portaudio\test\patest_saw.c"* dado ser o ficheiro de teste mais simples fornecido pela biblioteca.

Alteração da latência de saída

O PA tem tempos de latência baixos e altos predefinidos para o *streaming* de entrada e/ou saída de som. Os valores predefinidos são 200 ms (latência baixa) e 400 ms (latência alta). Um dos métodos para alterar a latência no PA é alterar esse valor na função *Pa_OpenDefaultStream*() que se encontra no ficheiro *pa_front.c* (*portaudio\src\common*). A linha a alterar dentro da função é:

hostApiOutputParameters.suggestedLatency =
Pa_GetDeviceInfo(hostApiOutputParameters.device)->defaultHighOutputLatency;

Para o seguinte:

hostApiOutputParameters.suggestedLatency = <valor_pretendido>;

Para mais informações consultar a página do PortAudio[PortAudio, 2008].

FFTW 3.2

A FFTW (*The fastest Fourier Transform in the West*) é uma biblioteca *OpenSource* escrita em C, desenvolvida pelo MIT, para calcular a transformada discreta de Fourier (DFT) em uma ou mais dimensões, com matrizes de tamanho arbitrário, podendo ser os dados reais e/ou complexos.

Algumas características do FFTW são:

- Velocidade. Suporte para código SIMD(*Single Instruction Multiple Data*) SSE(*Pentium III*+) SSE2 (Pentium IV+), AltiVec (PowerPC G4+) ou MIPS PS.
- Transformadas unidimensionais e multidimensionais.
- Transformadas rápidas com valores de entrada e saída puramente reais (sem recorrer a valores complexos).
- Portabilidade para todas as plataformas que contenham um compilador de C.
- Interface em C e em *Fortran*

Compilação da FFTW com o VC++

Os passos seguintes mostram como é compilada a FFTW com o VC++:

1- Descarregar o FFTW.

2- Dentro da raiz da biblioteca, existe um ficheiro com o nome "*config.h.in*". Copia esse ficheiro e altera o nome do mesmo para "*config.h*". Neste ficheiro existem por volta de 100 opções que são necessárias inserir (inicialmente definidas com um *#undef*) para definir as várias opções e características do compilador. A maioria das mesmas estão documentadas e é bastante intuitivo compreender o que definem. Na maioria das opções é apenas necessário fazer um *"#define*" a 1, embora algumas opções necessitam de um valor em particular (como por exemplo, a definir o tamanho dos dados). No CD entregue encontra-se um exemplo de como essas variáveis devem ser inseridas

3- Criar um novo projecto em VC++. Dependendo do objectivo final, ou podemos criar um ficheiro *dll* da biblioteca através do novo projecto, ou então, se preferirmos, seguir os passos seguintes num projecto já existente, compilando assim o FFTW já dentro do projecto no qual se pretende usar a mesma.

4- Inserir nos *paths* do projecto:

Kernel;
dft;
dft/scalar;
dft/scalar/codelets;
rdft, rdft/scalar;
rdft/scalar/r2cf;
rdft/scalar/r2cb;
rdft/scalar/r2r;
reodft;
api.

Se o suporte a **SIMD** foi definido no config.h é também necessário inserir os paths simd, simd/nonportable, dft/simd, dft/simd/codelets

5- Inserir os ficheiros '*.c' contidos nos paths inseridos no projecto

VTK

O *VTK* (*Visualization Toolkit*) é um pacote de *software* aberto (*open-source*) para computação gráfica, visualização e processamento de imagem. É orientado a objectos e suportado em várias plataformas. Implementado em *C++*, o *VTK* também pode ser utilizado com as linguagens *TCL*, *Python* e *Java*, prestando-se assim ao desenvolvimento de aplicações complexas, criação rápida de protótipos de aplicações e *scripting*. Embora não forneça componentes de interface de utilizador, pode ser integrado com bibliotecas tais como *Tk*, *X/Motif*, *JAVA*, *Visual Studio*, ou *Qt*. O *VTK* admite uma variedade de representações de informação, incluindo conjuntos desorganizados de pontos, informação poligonal, imagens, volumes e grelhas estruturadas, rectilíneas e não estruturadas. Oferece também filtros que permitem operar sobre estes vários tipos de informação [Ávila, 2004].

2 Hardware

InterTrax2

O *InterTrax* é um sensor de orientação com 3 ângulos de liberdade(*yaw, pitch e roll*). Desenhado para ser implementado de forma relativamente fácil em *Head Mounted Displays* para assim criar aplicações de realidade virtual e/ou aumentada de uma forma relativamente fácil.

O *InterTrax* usa uma combinação de oito sensores extremamente avançados que fornece a uma medição bastante fiável da velocidade angular. Os sinais dos sensores são analisados continuamente por um micro processador *on board* que usa vários algoritmos sofisticados para calcular a orientação do *tracker* no espaço. Como todos os cálculos são efectuados pelo microprocessador, não existe nenhum processamento extra no computador a que o sensor se encontra ligado.

i-glasses SVGA Head Mounted Display (HMD)

Os *i-glasses* ***SVGA*** *Head Mounted Display (HMD)* são constituídos por dois monitores, portátil, de alta resolução que podem ser usados por pessoas com problemas de visão, não sendo necessário remover os óculos de receita médica. Os óculos são formados por dois monitores LCD (*Liquid Crystal Display).*

Anexo 2 Experiências intermédias

Experiência em Matlab

Este ponto descreve a primeira experiência, efectuada em MATLAB (ver anexo 1, ponto 1) onde foram dados os primeiros passos no cálculo da auralização. Todos os valores são calculados previamente não permitindo assim ao utilizador que interaja com o mundo. As HRTF que foram usadas durante as experiências seguintes são as HRTF descritas no ponto. Esta experiência foi gerada em diferentes fases, cada uma com um propósito diferente.

a) Posicionamento de fontes sonoras em pontos fixos do espaço

Na primeira fase da experiência foi criado um algoritmo para posicionar várias fontes sonoras em pontos fixos do espaço a uma distância fixa do utilizador. O utilizador pode escolher o número de fontes sonoras que pretende activar e qual a posição aparente de cada uma (através da inserção dos ângulos azimute e elevação). Os ficheiros de som usados foram criados com auxílio de programas de edição e criação de som. O objectivo consiste em criar um pequeno *loop* de som com alguns instrumentos, guardar os mesmos em ficheiros separados e reproduzir cada som em posições distintas no espaço. Todos os ficheiros gerados usam uma frequência de amostragem de 44100 Hz e têm a mesma duração. O algoritmo usado nesta fase encontra-se descrito na Figura 52.

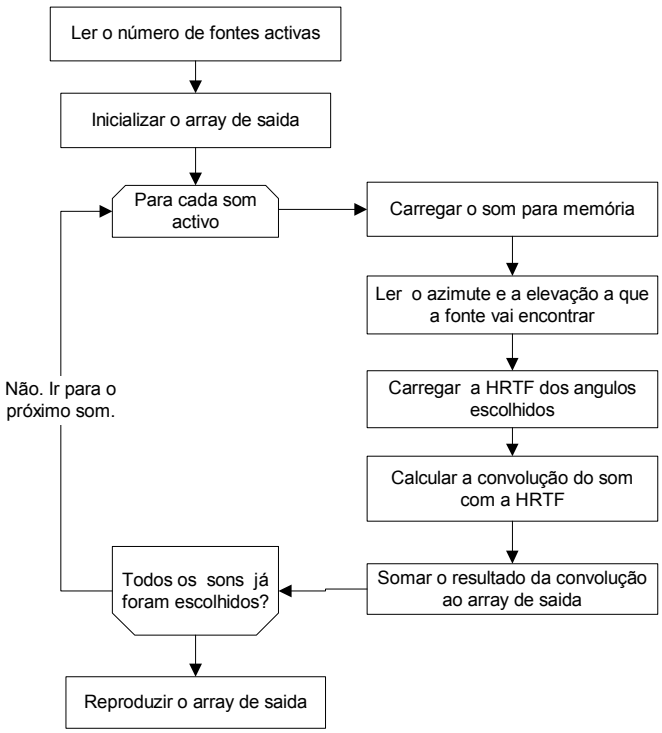


Figura 52 - Fluxograma do algoritmo de auralização

Como o som final resulta da soma de várias fontes sonoras, não está normalizado. A normalização consiste em determinar a amplitude máxima do som e atenuá-lo ar o mesmo tendo em conta esse valor. Para reproduzir um som de forma normalizada em *Matlab* é possível recorrer ao comando ***SoundSc*.** Através da implementação do algoritmo descrito em cima foi possível ter percepção da posição aparente da fonte sonora, dando assim o primeiro passo para a criação de um ambiente virtual 3D.

b) Inserção de movimento espacial

O objectivo da segunda fase da experiência consistiu na adição de movimento espacial a uma fonte sonora para assim fornecer ao utilizador a percepção que esta se encontra em movimento. A distância considerada entre o ouvinte e a fonte é constante. Esta fase foi subdividida em duas partes distintas. A primeira trabalha apenas com movimento no plano horizontal e a segunda no plano vertical.

c) Rotação num plano horizontal

Nesta fase a fonte sonora descreve uma trajectória circular num plano Horizontal (Azimute varia entre 0º e 360º). É dada a possibilidade ao utilizador de escolher o ângulo da elevação a que a fonte se encontra em relação ao ouvinte.

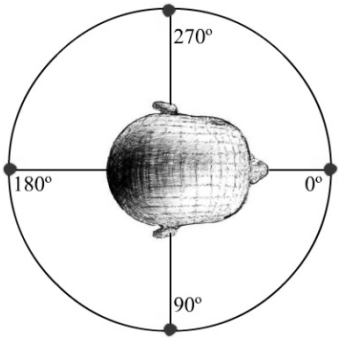


Figura 53 - Trajectória horizontal efectuada pela fonte sonora

Como o número de HRTF em cada elevação de ângulos é limitado e como não foram calculadas HRTF intermédias para os ângulos que não existem, o movimento da fonte sonora não é fluido, tendo o utilizador a percepção dos saltos entre cada HRTF.

Concluiu-se que o utilizador consegue ter percepção do movimento da fonte sonora. Verifica-se que durante as transições entre posições diferentes é gerado ruído (*clicks*) no som.

d) Rotação num plano Vertical

Nesta experiência a fonte sonora descreve uma trajectória onde a elevação varia entre 90º a -40º (elevação máxima e mínima para o qual existem HRTF medidas). A figura 52 ilustra o movimento efectuado.

O objectivo principal é verificar qual a percepção que o ouvinte tem do movimento vertical das fontes sonoras.

O som continua a não ser fluido, ocorrendo movimentos instantâneos entre as várias elevações. Quando numa elevação específica não existem amostras para o azimute inserido é usado o valor mais próximo deste.

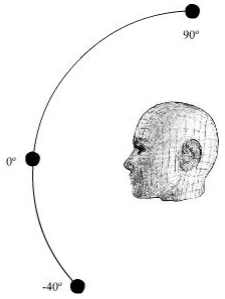


Figura 54 - Trajectória vertical efectuada pela fonte sonora

Verifica-se que o utilizador consegue ter uma vaga percepção que a fonte sonora está a mudar de posição. No entanto, não se consegue especificar com precisão qual o movimento descrito pela fonte. Tal reacção já era esperada dado que o ser humano, em espaço aberto, não tem uma boa percepção da elevação do som.

e) Adição de profundidade à fonte sonora

A parte final desta experiência consiste em compreender como o som é atenuado em espaço aberto, tendo em conta a distância entre a fonte sonora e o ouvinte.

O algoritmo criado desloca uma fonte sonora, com um azimute e elevação constante, entre distâncias diferentes em relação ao ouvinte. É dada assim a sensação de afastamento ou proximidade do som ao longo do tempo.

A atenuação a aplicar a cada amostra é igual à distância a que a fonte sonora se encontra do ouvinte quando esta amostra é reproduzida.

Para este teste o utilizador terá de inserir vários parâmetros. Além do azimute e elevação da fonte sonora, terá de inserir também a distância inicial e final do movimento e o tempo desejado para percorrer a distância definida. Verifica-se que o utilizador consegue identificar alterações na profundidade da fonte sonora em relação a si.

Reprodução de som 3D com o FMOD

O programa criado para conhecer as funcionalidades do *FMOD* pretende simular um ambiente sonoro 3D em tempo real. A biblioteca FMOD já tem suporte para a criação de som 3D, sendo extremamente fácil a criação de um ambiente virtual acústico com a mesma,

a) Criação de som 3D através do FMOD

Os ficheiros de som usados foram criados com auxílio de programas de edição e criação de som. O objectivo base era reproduzir 4 sons diferentes, cada um numa posição aparente no espaço. Os sons criados consistiam num pequeno ritmo de uma bateria simulada gravando cada parte da bateria (Bombo, Caixa, Prato de choques) em ficheiros distintos. O quarto som usado é uma pequena melodia cuja duração é quatro vezes maior que os sons da bateria. Um dos motivos para este som ser mais longo foi para verificar como o FMOD se comportava a sincronizar *loops* de som com tamanhos distintos.

De seguida está uma pequena descrição dos tipos de dados mais importantes da biblioteca FMOD.

> **FMOD_RESULT**- Todas as funções do FMOD retornam um valor para verificar se existiram erros na execução. Esse valor é do tipo FMOD_RESULT que é um tipo enumerado que contém todos os valores possiveis retornados por uma função.
>
> **FMOD_SYSTEM –** Para utilizar o FMOD é necessário a instanciação de um objecto deste tipo, ou seja, um objecto do Sistema. É a base de tudo o que FMOD faz ou permite fazer.

A seguir encontra-se uma breve descrição do trabalho efectuado:

A primeira passo consiste criar o sistema e inicializar o sistema. Isso é efectuado através das instruções seguintes:

```
FMOD_RESULT result;
FMOD_SYSTEM *system;
```

```
result = FMOD_System_Create(&system);
ERRCHECK(result);

result = FMOD_System_Init(system, 4, FMOD_INIT_NORMAL, NULL);
ERRCHECK(result);
```

Neste excerto de código está a ser criado o nosso sistema (*FMOD_System_Create*()) e a iniciar o mesmo com a forma mais básica que o FMOD permite (*FMOD_INIT_NORMAL*). Vão ser usados apenas 4 canais virtuais dado porque apenas vão ser reproduzidos 4 sons. A função *ERRCHECK()* é usada para verificar se não existiram erros na execução do programa.
Como o resultado vai ser reproduzido em auscultadores, define-se a saída como estéreo:

```
FMOD_System_SetSpeakerMode(system,FMOD_SPEAKERMODE_STEREO);
```

Agora, já com o sistema criado e inicializado é necessário abrir e inicializar os sons. Um exemplo do código necessário para abrir e inicializar um som é:

```
FMOD_SOUND *sound;
FMOD_System_CreateSound(system,
                        "d1.wav",
                        FMOD_SOFTWARE | FMOD_3D FMOD_LOOP_NORMAL
                        ,0
                        ,sound);

  FMOD_System_PlaySound(system, FMOD_CHANNEL_FREE, sound[0], TRUE,
  &channel[0]);
```

Com o comando ***FMOD_System_CreateSound*** carrega-se o ficheiro de som para memória e especifica-se que o som é para ser reproduzido em loop. A informação do som irá ser guardada na variável ***sound***. O ***FMOD_System_PlaySound*** reproduz um som num canal especifico. No exemplo dado o som vai ser reproduzido num canal que esteja livre. O som que vai ser associado a um canal é o som que está contido na variável *sound.* O parâmetro que está definido a **TRUE** serve para o som ser inicializado em pausa. Isso é útil quando queremos reproduzir vários sons ao mesmo tempo, para assim ser possivel iniciar todos os sons pretendidos antes da sua reprodução.

O próximo passo consiste em definir posições virtuais para cada canal. Para isso é criado um vector que define a posição em que cada fonte sonora irá estar no espaço. É possível também definir um vector velocidade para atenuar ou acentuar o efeito de Doppler, no entanto, o mesmo não foi considerado De seguida está um exemplo da forma que os Atributos 3D são inseridos em cada canal canal.

```
FMOD_CHANNEL *channel;
FMOD_VECTOR pos1 = { 0.0f, 0.0f, 1.0f };
result = FMOD_Channel_Set3DAttributes(channel, &pos1, 0);
```

Depois de todas as fontes estarem devidamente posicionadas é necessário definir os atributos do utilizador. A orientação do utilizador é definida pelos vectores Forward e Up e pela posição da câmara no espaço. O comando para definir os atributos do ouvinte é:

```
FMOD_System_Set3DListenerAttributes(system, 0,
                                    &pos,
                                    0,     //Vector velocidade
                                    &forward,
                                    &up);
```

O vector velocidade foi novamente deixado a zero por não estarmos a considerar o efeito de Doppler

b) Interacção com o utilizador

O último passo desta experiência explica como foi dada a possibilidade de interacção ao utilizador com o ambiente de RV. A interacção com o ambiente é feita recorrendo ao teclado e/ou usando o sensor Intertrax.

Como estamos perante uma aplicação em que o utilizador vê o sistema em primeira pessoa (a posição da câmara corresponde á posição do utilizador), sempre que a posição deste se altera é necessário actualizar os valores da câmara. Para alterar apenas a posição da câmara necessitamos de incrementar o seu vector de posição, ficando os vectores Up e Forward iguais. Quando alteramos um dos ângulos de rotação(yaw, pitch, roll), a posição da câmara não é alterada mas, no entanto, é necessário aplicar a rotação correcta aos vectores forward e Up.

c) Criação de informação visual

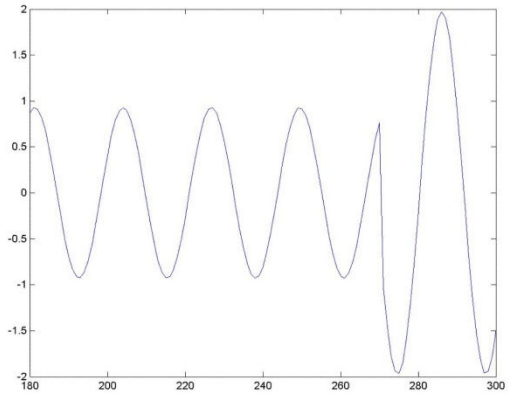
Para concluir a experiência foram feitos mais dois pequenos passos. O primeiro consistiu em criar nova janela de visualização recorrendo à biblioteca VTK. Nessa janela podemos ver uma esfera na posição aparente de cada som. Foi também seleccionada a opção de *fullscreen* da janela para permitir ao utilizador poder usar os óculos de realidade virtual (*HMD*).

d) Conclusões sobre a biblioteca *FMOD*

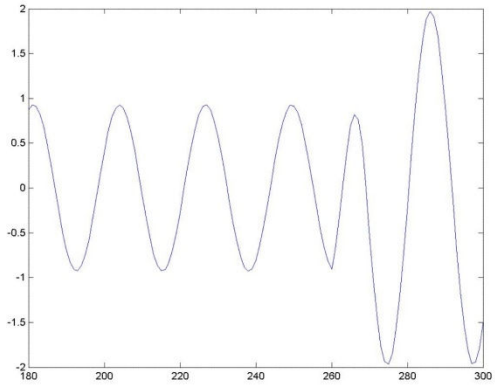
Através desta pequena demonstração com comprovar que o *FMOD* fornece uma forma relativamente fácil de simular som 3D sendo uma ferramenta extremamente útil para criar ambientes virtuais sonoros em tempo real. No entanto, o facto de o *FMOD* abstrair o utilizador da maior parte dos cálculos gerados e a forma de como o *buffer* de saída é preenchido, tornam o resultado de saída pouco maneável, ou seja, não é possível adicionar efeitos (reverberação por exemplo). Outro dos problemas consiste no facto de o *FMOD* não explicar qual o algoritmo usado para gerar o som 3D. Assim, torna-se complicado tentar simular a realidade devido às limitações apontadas anteriormente e decidiu-se assim optar por outra biblioteca.

Anexo 3 Imagens da atenuação da interpolação

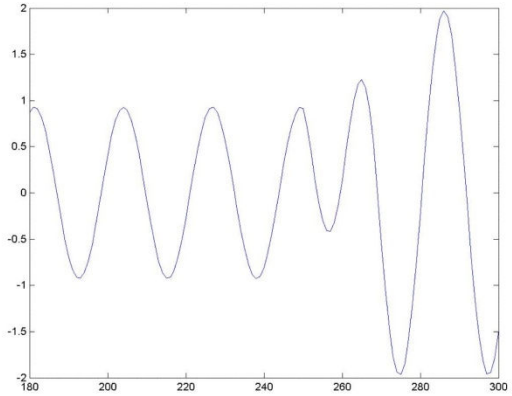
As imagens que se seguem mostram a atenuação dos artefactos gerados entre transições de HRTF diferentes usando um número variável de amostras na interpolação. As primeiras duas páginas correspondem ao ouvido esquerdo e as duas seguintes ao ouvido direito.



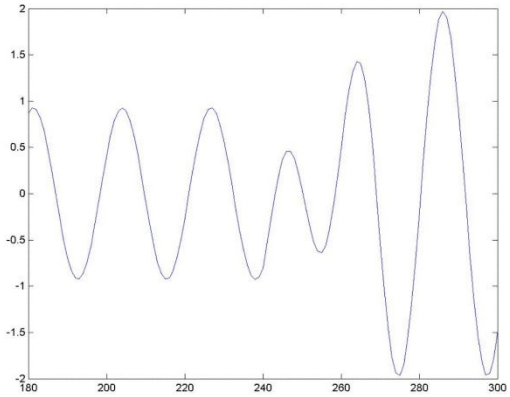
Amostras = 0



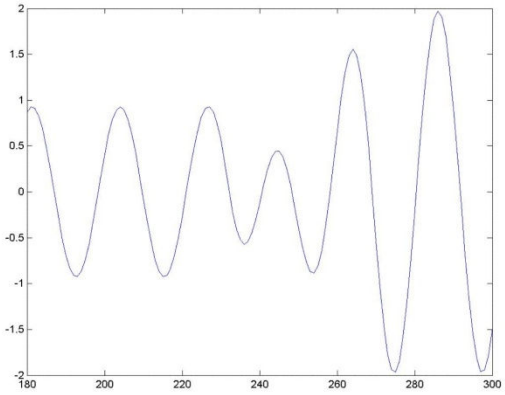
Amostras = 10



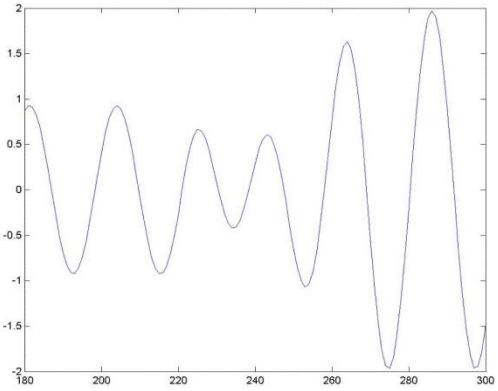
Amostras = 20



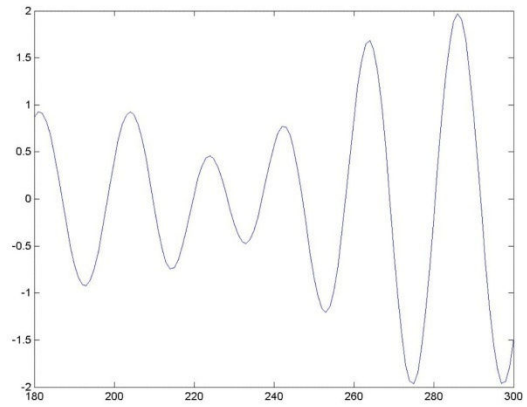
Amostras = 30



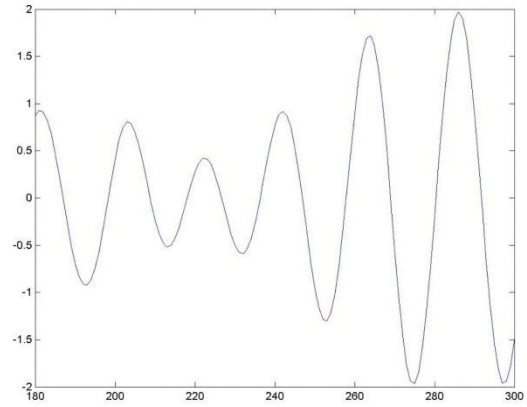
Amostras = 40



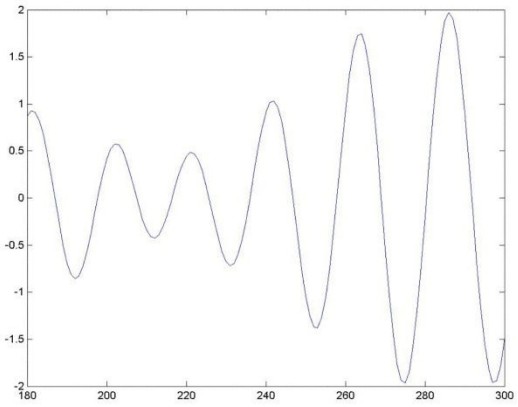
Amostras = 50



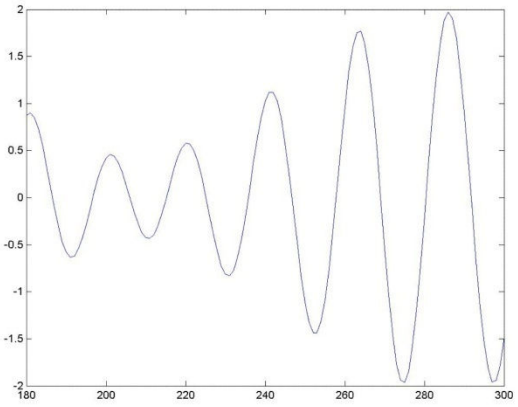
Amostras = 60



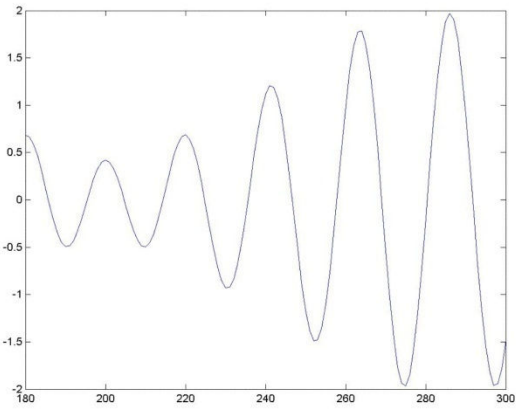
Amostras = 70



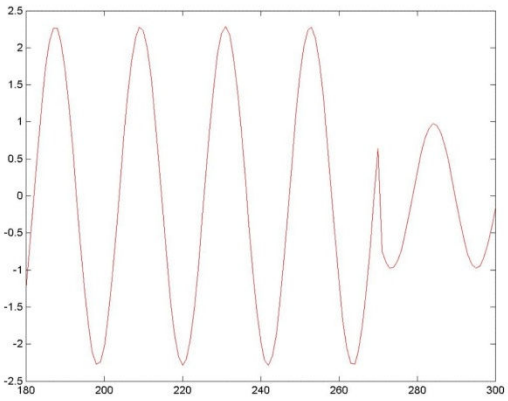
Amostras = 80



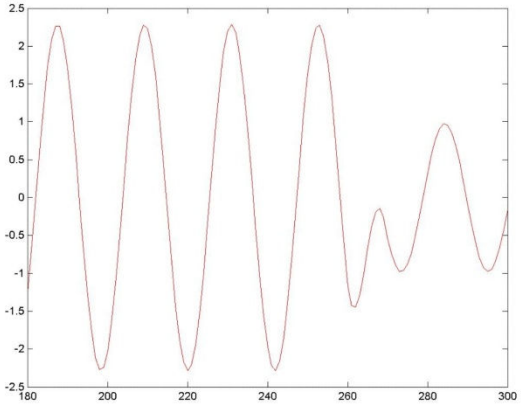
Amostras = 90



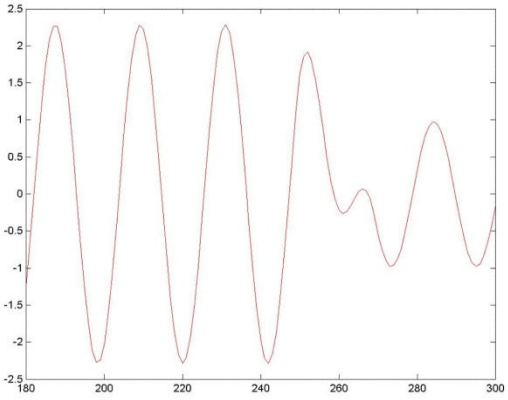
Amostras = 100



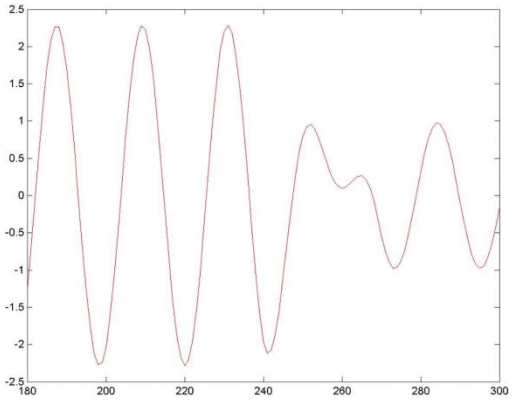
Amostras = 0



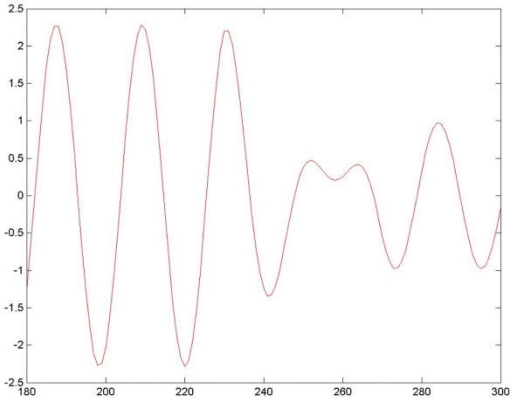
Amostras = 10



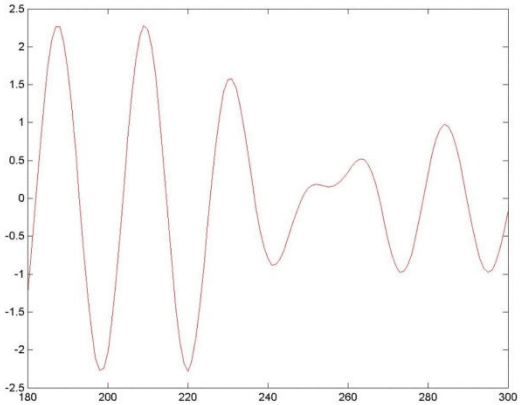
Amostras = 20



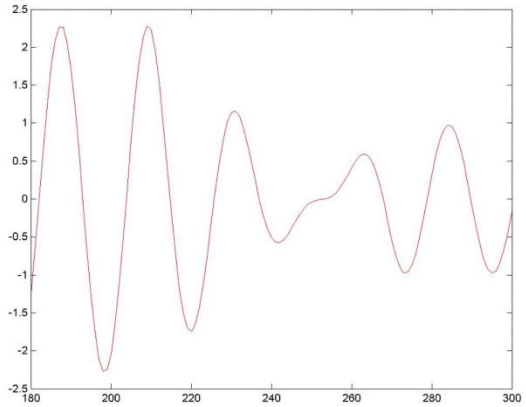
Amostras = 30



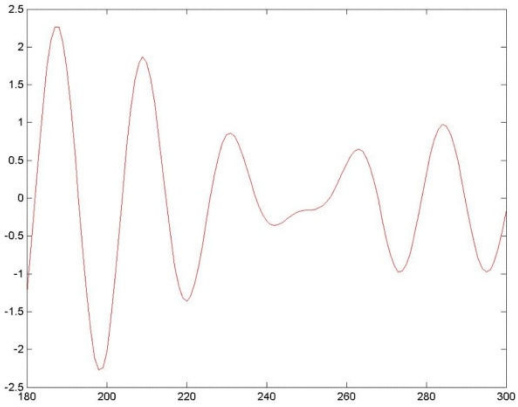
Amostras = 40



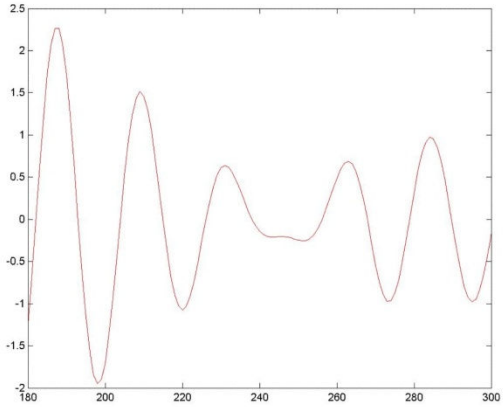
Amostras = 50



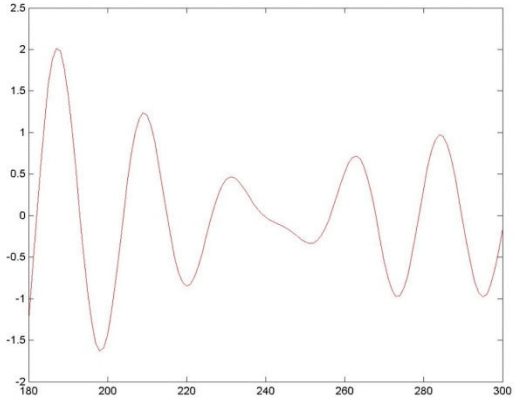
Amostras = 60



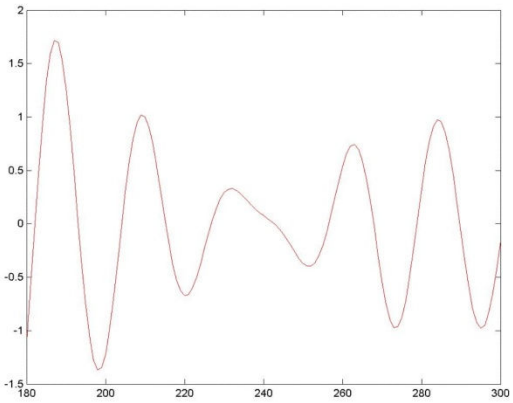
Amostras = 70



Amostras = 80



Amostras = 90



Amostras = 100

7 Referências

[Allen e Berkley, 1979]: Allen, J. B. and Berkley, D. A. "*Image Method for Efficiently Simulating Small-Room Acoustics*", J. Acoust. Soc. Am., vol. 65, 1979, 943-950**.**

[ASIO, Junho 2008]: WWW - *"ASIO SDK*".

URL (Junho 2008): http://www.steinberg.de/324_1.html

[Ávila, 2004]: Ávila, Lisa S., "*The VTK User's Guide*", Kitware, 2004.

[Aurora, 2008]: WWW - "*Aurora"*.

URL (Junho 2008): http://www.aurora-plugins.com/Aurora_XP/index.htm

[Begault, 1994]: Begault, Durand R. "*An Introduction to 3-D Sound for Virtual Reality*", NASA-Ames Research Center, Moffett Field, CA, 1992.

[Beranek, 1954]: Beranek , Leo "*Acoustics"*, McGraw-Hill, 1954.

[Casaleiro et al, 2007]: Casaleiro R., Seco R., Campos G., Dias P, Sousa Santos B. *"Sala de Espectáculos Virtual: Articulação em tempo real dos Ambientes Visual e Acústico"*. Actas do 15º Encontro Português de Computação Gráfica. Microsoft, Tagus Park, Porto Salvo. 15-17 Outubro 2007. 51-57.

[Dahl et al, 2000]: Dahl, Luke & Jot ,Jean Marc, "*A reverberator based on absorbent all-pass filters",* COST G-6 Conference on Digital Audio Effects (DAFX-00), 2000, Verona, Italy.

[Dias et al, 2008]: Dias P., Campos G., Santos V., Casaleiro R., Seco R., Sousa Santos B. "*3D Reconstruction and Auralization of the "Painted Dolmen" of Antelas"*. In Proceedings of the Electronic Imaging 2008 conference, SPIE Vol. 6805, 6805OY, Three-Dimensional Image Capture and Applications 2008, San Jose, California, USA. 28-29 Janeiro 2008.

[DirectSound, 2008]: WWW - *'DirectSound API'*.

URL (Junho 2008): http://msdn.microsoft.com/en-us/library/bb219818(VS.85).aspx

[Everest, 2001]: Everest, F. Alton "*The master handbook of acoustics"*, McGraw-Hill, 2001.

[Farina, 1993]; Farina, A. "*Convolution of anechoic music with binaural impulse response*" Proc. of PARMA-CM Users Meeting, 1993, Parma.

[Farina, 1995a]: Farina, A. "*Pyramid Tracing vs. Ray Tracing for the simulation of sound propagation in large rooms*" - COMACO95, Proc. of Int. Conf. on Comput. Acoustics and its Environmental Applications, 1995, Southampton, England.

[Farina, 1995b]: Farina, A. "*RAMSETE - a new Pyramid Tracer for medium and large scaleacoustic problems*", Proc. of Euro-Noise 95, Lyon, France 21-23 Março 1995.

[FMOD, 2008]: WWW – "*FMOD sound system"* .

URL (Junho 2008): http://www.fmod.org

[FFTW, 2008]: WWW – *"FFTW- The fastest Fourier Transform of the West"* .

URL (Junho 2008): http://www.fftw.org

[Gardner, 1998]: Gardner, William "*Chapter .3 Reverberation Algorithms*", em Kahrs, M. and Brandenburg, K. Editors. "*Applications of Digital Signal Processing to Audio and Acoustics"*. Kluwer Academic Publishers. 1998.

[Gardner, 1999]: Gardner, William "*3D Audio and Acoustic Environment Modeling"*, Wave Arts, Inc., 1999.

[Howard et al, 1996]: Howard, D. M. and Angus, J. A. S. "*Acoustics and Psychoacoustics"*, Oxford: Focal Press.Harmony Central, 1996.

[HRTF, 2008]: WWW '*HRTF Measurements of a KEMAR Dummy-Head Microphone* '.
URL (Junho 2008): http://sound.media.mit.edu/KEMAR.html

[Kendall, 1995]: Kendall, Gary S. "*A 3-D Sound Primer: Directional Hearing and Stereo Reproduction"*, Computer Music Journal, 19(4), 1995, 23-46.

[Libsndfile, 2008]: WWW - "*libsndfile"*.
URL (Junho 2008): http://www.mega-nerd.com/libsndfile/

[Matlab, 2008]: WWW - "*The MathWorks - MATLAB and Simulink for Technical Computing*".
URL (Junho 2008) www.mathworks.com

[Moorer, 1979]: Moorer J. A. "*About this Reverberation business"*, Comput. Music J., vol. 3, no. 2, 1979, 13–18.

[Moorer, 1987]: Moorer, J. A. "*About This Reverberation Business*", Foundation of Computer Music, The MIT press, 1987, 605-639.

[Orfanidis, 1996]: Orfanidis, Sophocles J. ,"*Introduction to Signal Processing",* Prentice Hall, 1996.

[Park e Chio, 2006]: Young-cheol Park, Taik-sung Chio, Jae-woong Jung Dae-hee Youn, Siwook Nam, and Jungmin Song, "*Low Complexity 3D Audio Algorithms for Handheld Devices*", AES, 2006, Seoul, Korea.

[Portaudio, 2008]: WWW "*PortAudio an Open-Source Cross-Platform Audio API"*.
URL (Junho 2008):http://www.portaudio.com

[Rochesso, 1997]: Rochesso, Davide, "*Maximmaly diffusive, Yet Efficient Feedback Delay Networks for artificial reverberation"*, IEEE signal processing letters, vol. 4, nº 9, 1997.

[Savioja, 1999]: Savioja, L; Huopaniemi, J; Lokki, T & Väänänen, R ,"*Creating Interactive Acoustic Environments"*, J. AES Vol 47, No 9, 1999

[Schafer et al, 1998]: McClellan, Schafer, & Yoder, "*DSP First: A Multimedia Approach"*, New Jersey: Prentice Hall, 1998.

[Schroeder, 1961]: Schroeder, M.R. *"Colorless Artificial Reverberation",* J. Audio Eng. Soc., vol. 9, 1961, 192-197.

[Smith, 2008]: Smith, J.O. *"Physical Audio Signal Processing"*, August 2007 Edition, online book.

URL (Junho 2008): http://ccrma.stanford.edu/~jos/pasp.

[Vickers, 2006]: Vickers, Earl; Wu, Jian-Lung & P. G. Krishnan, "*Frequency Domain Artificial Reverberation using Spectral Magnitude Decay", Proceedings of the AES 121st Convention*, Preprint 6926, 2006, San Francisco.

[VS2005, 2008]: WWW – "*Visual Studio 2005 Developer Center"*.

URL (Junho 2008): msdn.microsoft.com/vstudio/

[White, 1975]: White, Frederic A. "*Our acoustic Environment",* John Wiley & Sons, 1975

[Wu e Mu, 1997]: Jiann-Rong Wu , Cha-Dong Duh , Ming Ouhyoung , Jei-Tun Wu, *Head motion and latency compensation on localization of 3D sound in virtual reality*, Proceedings of the ACM symposium on Virtual reality software and technology, Setembro 1997, 15-20, Lausanne, Switzerland.

[Zölzer, 2002]: Zölzer, Udo, "*DAFX: Digital Audio Effects",* John Wiley & Sons, Inc., 2002